
Client/Server : Connection Establishing and Configuration

By

Loo Chee Keong (WEK 990042)

Under Supervision of

Pn. Hannyzzura Pal @ Affal

&

Encik Mohd. Nizam Ayub

Under Moderation of

Cik. Nurul Fazmidar

*This project is submitted to the Faculty of Computer Science and
Information Technology, University of Malaya in partial fulfillment of the
requirements for the Bachelor of Computer Science (Honors)*

Session 2001/2002

ABSTRACT

This report will firstly cover the introductory of the client/server architecture. In this part, I will try to give a full definition of the client/server architecture. Secondly, this report will review the literature. In this part, I started with the basic knowledge about the client/server technologies. At the end, I have discussed about the analysis on some other existing system models such as the Centralized Multi-User Architecture and Distributed Single-User Architecture.

The first chapter of the Methodology part is describing the methods that are suggested and have been used to define client/server architecture and to develop a client/server system model. The next chapter in this part covers the system and application design, module design and interface design. Finally, in Implementation Part, I will show the ways to establish a client-server connection to Oracle Database Server with Windows 2000. Also stated, will be the implementation and testing for this application called the **"FSKTM Multimedia Database"**. Finally, a brief and attractive User Manual will be stated in the last chapter.

ACKNOWLEDGEMENT

I would like to express sincere appreciation to Pn. Hannyyzzura Pal for her advice and guidance during our research. I would also like to thank my moderator, Cik. Nurul Fazmidar and my thesis member, Gan Hong Fee for his help and hard work. Thank to all my friends who have given me a helping hand sincerely and not to forget my parents and family members for their support and care. Thanks!

TABLE OF CONTENTS

ABSTRACT	i
ACKNOWLEDGEMENT	ii
TABLE OF CONTENTS	iii
LIST OF FIGURE	vi

PART I INTRODUCTION

Chapter 1 Introduction	1
1.1 Introduction to Client/Server	2
1.1.1 What is Client/Server Architecture?	2
1.1.2 Client/Server as a Design Approach	3
1.1.3 Client/Server Database	5
1.2 System Objectives	6
1.3 Advantages of Client/Server Database	7
1.3.1 Flexibility	7
1.3.2 Scalability	8
1.3.3 Technology Potential	9
1.3.4 Cost	10
1.4 Scope of Project	11
1.5 Gantt Chart	12

PART II Literature Review

Chapter 2 Client Architecture	13
2.1 Application and Tools	13
2.1.1 End-User Tools	14
2.1.2 Developer Tools	15
2.2 Operating System	17
2.3 Graphical-user Interface (GUI) Facilities	17
2.4 Hardware Platform	18
2.5 Database Access	19
2.5.1 Embedded SQL	20
2.5.2 Function call interface	22
2.6 Middleware	23
2.7 Interprocess Communication Protocols	23
2.7.1 Named Pipes	24
2.7.2 Remote-procedure calls	25
2.7.3 Application Program-to-Program Communication (APPC)	26
Chapter 3 Network Technologies	27
3.1 Local Area Network (LAN)	27
3.1.1 Ethernet	28

3.1.1.1 Topology	29
3.1.1.2 CSMA/CD	31
3.1.2 Token Ring	32
3.1.2.1 Topology	32
3.1.2.2 Token Passing	33
3.2 The OSI Model	34
3.2.1 Layer 1: Physical Layer	36
3.2.2 Layer 2: Data-link Layer	37
3.2.3 Layer 3: Network Layer	37
3.2.4 Layer 4: Transport Layer	38
3.2.5 Layer 5: Session Control Layer	38
3.2.6 Layer 6: Presentation Layer	39
3.2.7 Layer 7: Application Layer	40
3.3 Node Components	40
3.3.1 DB-interface	41
3.3.2 Network protocol	42
3.3.3 Network adapter driver	43
3.3.4 Network adapter card	44
3.4 Network Connectivity Components	45
3.5 Media	46
3.5.1 Coaxial Cable	46
3.5.2 Twisted-pair Cable	47
3.5.3 Fiber-optic Cable	48
3.5.4 Repeaters	49
3.5.5 Hubs	49
3.5.6 Concentrators	49
3.5.7 Bridge	50
3.5.8 Router and Brouter	51
3.5.9 Gateway	52
Chapter 4 Server Architecture	53
4.1 Server Operating System	56
4.2 Server Hardware	59
4.3 Database Access	60
4.4 Distributed Data Access	61
Chapter 5 Security in Client/Server	65
5.1 User Authentication	66
5.1.1 Database Authentication	67
5.1.2 External Authentication	70
5.1.3 Enterprise Authentication	72
5.2 Data Encryption	73
5.2.1 Symmetric Key Cryptography	73

5.2.2 Public Key Cryptography	75
5.2.2.1 RSA Algorithm	76
5.2.2.2 Example Scenario for Digital Signature	79
5.3 Authentication	81
5.3.1 Authentication Protocol	81
Chapter 6 Analysis on Existing System Model	83
6.1 Centralized Multi-User Architecture	83
6.1.1 The Strengths of Centralized Multi-User Architecture	84
6.1.2 The Weakness of Centralized Multi-User Architecture	85
6.2 Distributed Single-User Architecture	86
6.2.1 The Strengths of Distributed Single-User Architecture	88
6.2.2 The Weakness of Distributed Single-User Architecture	89

Part III Methodology

Chapter 7 Methodologies	91
7.1 Methodologies Used	91
7.2 Solution for System Definition	94
7.2.1 Process reengineering	94
7.2.2 System Development	96
7.2.3 Architecture	98
7.3 Solution for System Development	99
7.3.1 Process reengineering	99
7.3.2 System Development	100
7.3.3 Architecture	101
Chapter 8 System Design	104
8.1 System and Application Design	104
8.2 System Architecture	105
8.3 Module Design	111
8.3.1 Data Retrieving Module	111
8.4 User Interface Design	112
8.5 Designing the Effective	114
8.6 Problems Faced	115
8.7 Estimation on Project Output	116

Part IV Implementation

Chapter 9 Configuring Client and Server	117
9.1 Setting Up Client in Windows 2000	117
9.2 Setting Up Server in Windows 2000	120
9.3 Installing The Oracle Client	120
9.4 Configuring The Oracle Client	122

9.4.1 Local Naming	123
9.5 Configuring The Oracle Server	126
Chapter 10 Implementation and Testing	134
10.1 Development Environment	134
10.1.1 Hardware Requirements	134
10.1.2 Tools For System Development	134
10.1.3 Software Tools For System Development	135
10.1.4 Software Tools For Documentation	135
10.2 Program Development	135
10.2.1 Review The Program Documentation	136
10.2.3 Design The Program	136
10.2.3 Code The Program	136
10.2.4 Test The Program	136
10.2.5 Document The Program	137
10.3 Program Coding	137
10.3.1 Coding Approach	137
10.3.1.1 Coding Style	137
10.3.1.2 Code Documentation	137
10.3.2 Database Connection	138
10.3.2.1 Connection Object	140
10.3.2.2 Retrieving Data Via OraDynaSet	140
10.3.2.3 Using The OraSqlStmnt	140
10.3.2.4 OraBFile	141
10.4 System Testing	144
10.4.1 Unit Testing	145
10.4.2 Integrating Testing	145
10.4.3 System Testing	145
Chapter 11 System Evaluation	147
11.1 Problems Faced and Solution	147
11.1.1 Lack Of Knowledge In Establishing Connection	147
11.1.2 Lack Of Knowledge In Programming Language	147
11.1.3 Wide Area Of Studies	148
11.2 Strengths Of The System	148
11.2.1 Simple and Ease of Use Graphical User Interface	148
11.2.2 System Transparency	148
11.2.3 Allow Searching of Data	148
11.2.4 Allow Sorting of Record by Entity Type	148
11.3 System Limitation	149
11.3.1 Disadvantages of Using OraBFile	149
11.3.2 Limited Data Format That Can Be Read	149
11.4 System Enhancement	149
11.4.1 Change OraBFile to OraBlob or OraClob	149

11.4.2 Login Form	150
11.4.3 Integrate Plug_Ins Into Visual Basic	150
Chapter 12 User Manual Of FSKTM Multimedia Database	151
12.1 Getting Started: The First Page	151
Chapter 13 Conclusion	161
References	162
Books	162
World Wide Web	164
Appendix	165

LIST OF FIGURES

<i>Figure 1.1 Client/Server Architecture</i>	3
<i>Figure 1.2 Client and Server Roles</i>	4
<i>Figure 1.3 A Client/Server Database</i>	6
<i>Figure 1.4 Client/Server Architecture Performance Characteristics</i>	8
<i>Figure 2.1 Client Components</i>	13
<i>Figure 2.2 Database Connectivity</i>	20
<i>Figure 3.1 Standard Ethernet Bus Topology</i>	30
<i>Figure 3.2 Typical 10Base Topology</i>	30
<i>Figure 3.3 Typical Token Ring Topology</i>	33
<i>Figure 3.4 OSI Architecture</i>	36
<i>Figure 3.5 DB-Net Interface In Client/Server Model</i>	42
<i>Figure 4.1 Dedicated Versus Non-Dedicated</i>	55
<i>Figure 4.2 Server Operating System</i>	55
<i>Figure 4.3 Server To Server Data Access</i>	62
<i>Figure 5.1 Symmetric Key Cryptography</i>	74
<i>Figure 5.2 Public Key Cryptography</i>	76
<i>Figure 5.3 RSA Public Key Cryptography</i>	77
<i>Figure 5.4 Digital Signature Process</i>	78
<i>Figure 5.5 Creating A Digital Signature For A Document</i>	79
<i>Figure 5.6 Protocol ap 5.0 Scenario</i>	82
<i>Figure 6.1 Centralized Multi-User Architecture</i>	83
<i>Figure 6.2 Isolated Single-User Architecture</i>	86
<i>Figure 6.3 File Server Architecture</i>	87
<i>Figure 7.1 Methodology Phases</i>	92
<i>Figure 7.2 The Natural Cycle</i>	94
<i>Figure 8.1 3-tier and 2-tier Client/Server Architecture</i>	106
<i>Figure 8.2 System Model</i>	109
<i>Figure 8.3 Data Flow Diagram (DFD)</i>	112
<i>Figure 8.4 Login Form</i>	113
<i>Figure 8.5 Administrator Query Form</i>	113
<i>Figure 8.6 User Query Form</i>	114
<i>Figure 9.1 Local Area Connection Status Form</i>	117
<i>Figure 9.2 Choosing Features For New Connection</i>	118
<i>Figure 9.3 Select The Network Component Type</i>	118
<i>Figure 9.4 Initialize The IP Address and Subnet Mask</i>	119
<i>Figure 9.5 Oracle's Universal Installer</i>	120
<i>Figure 9.6 Specifying The Location For The Connection</i>	121
<i>Figure 9.7 Choosing The Installation Type</i>	121
<i>Figure 9.8 Net8 Assistant</i>	122
<i>Figure 9.9 Adding The Naming Method for Oracle Client</i>	124
<i>Figure 9.10 Configuring the Local Naming Method</i>	125
<i>Figure 9.11 Security Features In Net8</i>	126
<i>Figure 9.12 Listener Configuration</i>	127

Figure 9.13 Add A New Listener	128
Figure 9.14 Naming The Listener	129
Figure 9.15 Select Protocol	129
Figure 9.16 Select Port Number 1521	130
Figure 9.17 Set The Listener To Listen From Client Computer, CheeKeong	131
Figure 9.18 Set The Listener To Listen From Service Name, FSKTM	132
Figure 9.19 Testing The Connection Between Oracle Server and Oracle Client	133
Figure 10.1 The Five Steps Of Program Development	135
Figure 10.2 Testing Stages	144
Figure 12.1 FSKTM Multimedia Database Main Menu	151
Figure 12.2 Student's Details Form	152
Figure 12.3 Add Images Form	154
Figure 12.4 Add Doc's ID Form	155
Figure 12.5 View Image Form	156
Figure 12.6 Search Ids Form	157
Figure 12.7 Display Image Form	158
Figure 12.8 Listen Audio Form	159
Figure 12.9 Play Audio Form	159
Figure 12.10 Common Dialog Box	160

Chapter 1 Introduction

Each student of the Faculty of Computer Science and Information Technology, University Malaya is required to undergo this final year project paper which consists of Projek Ilmiah Tahap Akhir I (WXES 3181) and Projek Ilmiah Tahap II (WXES 3182).

So, my thesis partner Mr. Gan Hong Fee and I have chosen a project titled **"A Client-Server Multimedia Database for FSKTM Student"**, as our final year project question. This question is a question provided from our supervisor **Pn. Hannyzzura Pal @ Affal**.

On 28th August 2001, a report named, **Client/Server : Connection Establishing and Configuration**, a Proposal of Thesis Project has been submitted to my supervisor **Pn. Hannyzzura Pal @ Affal** and moderator **Cik. Nurul Fazmidar**. This report has stated a lot of comprehensive information of establishing Client/Server connection. Also, system design module of Data Retrieving has been discussed.

Now that, we have already finished the project and this report is written to finalize the project. This report will be submitted to **Pn. Hannyzzura Pal @ Affal** and **Cik. Nurul Fazmidar** by 25 January 2002

We will start our discussion with the Introduction to Client/Server definition, which this might lead us to a better understanding of why we are using the Client/Server System Model.

1.1 Introduction to the Client-Server

Let me start this thesis report, by giving a clear definition about the *Client-Server Architecture* before we further approach to the definition of *Client-Server Database*.

1.1.1 What is a Client-Server Architecture?

From the book "*Client/Server System Design & Implementation* by *Larry T. Vaughn*", there is clearly stated in that book the definition of the Client/Server Architecture is:

"Client/server architecture is an application design approach that results in the decomposition of an information system into a small number of server functions, executing on one or more hardware platforms, that provide commonly used services to a larger number of clients functions, executing on one or more different but interconnected hardware platforms, that perform more narrowly defined work in reliance on the common services provided by the server functions."

Even with this full definition and despite some large numbers of articles and seminars on this subject, there is still confusion in many information system professionals' minds regarding the application of this approach and its impact in the organizations using more traditional architectures. Like art, where a few can agree on the definition but everyone knows it when they see it, "client/server architecture" means different things to different people and depending on their perceptions and backgrounds.

1.1.2 Client/Server Architecture as a Design Approach

This approach designs an application so that the functional components of an application are partitioned in a manner that allows them to be spread across, and executed on, multiple different computing platforms sharing access to one or more common repositories of data (Figure 1.1).

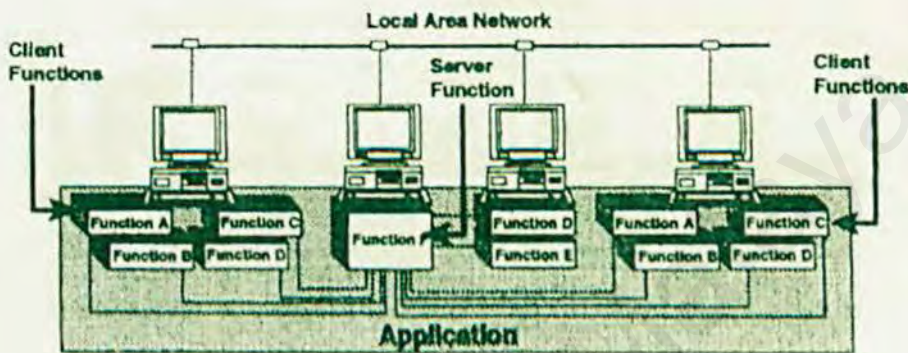


Figure 1.1 : Client/Server Architecture

Client/server architecture is therefore a design approach that distributes the functional processing of an application across two or more different processing platforms. The phrase "client/server" reflects the roles played by the application's functions as they interact with one another. One or more of these functions provide a service, most commonly a database server, which is commonly used by other functions across the application(s). When providing such service, these functions are identified as playing a "server" role in the application. Functions that request a "service" from another are playing the "client" role within the application. One thing that is to be highlighted here, is we have to remember that it is quite possible for a specific function to play both the "client" role and also the "server" role in terms of its interaction with other functions. For

example, let us consider the Sybase's Open Server or IBM's DRDA (Distributed Relational Data Architecture) facilities. In these instances, the database server plays the role as a gateway to another database server (client's role) and at the same time plays its own role as a database server (Figure 1.2).

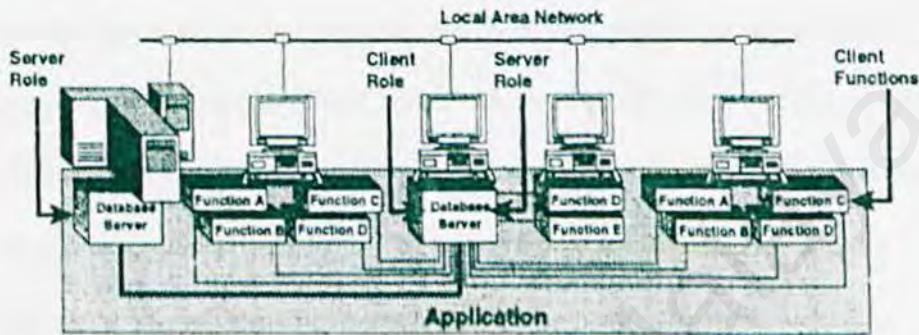


Figure 1.2 : Client and Server Roles

Actually, client/server application is most commonly applied in a database system but any applications that decompose its functions in a manner that play client and server roles across different platform boundaries can be also accurately described as a client/server application. Examples of currently available client/server applications include electronic-mail (e-mail) applications that feature administration and store/forward server functions, shared networked communication servers and shared network fax servers. Workgroup applications, that provided shared communications, scheduling, workflow routing and a variety of other functions are also being built on a client/server architectural foundation.

1.1.3 Client/Server Database

Now that we are clear about the definition of the Client/Server Architecture, let us further approach to the definition of the Client/Server Database. Referring from the book *"Client/Server Databases (second edition) by Joe Salemi"*, Client/Server Database splits the database processing between two systems: the client PC which runs the database application and the database server which runs all or part of the actual DBMS. The LAN File Server continues to provide shared resources, such as disk space for applications and printers. The database server can run on the same PC as the File Server, or (as in more common) on its own computer. The database application on the client PC, referred to as the front-end system, handles all the screen and user input/output processing. The back-end system on the database server handles data processing and disk access. For instance, when a user sends a request for data that is stored in the database server, the front-end application that can be built from various kinds of programming language will send this request through the network to the server. The database server will then performs the actual search and sends back only the data that match the user's query (Figure 1.3).

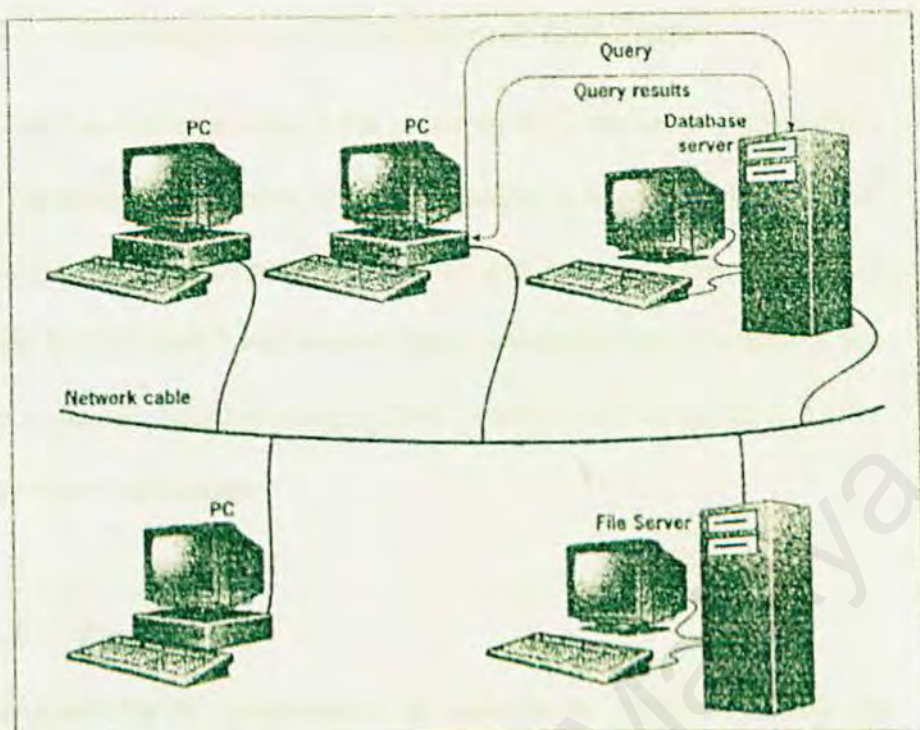


Figure 1.3 : A Client/Server Database

1.2 System Objectives

To implement and configure a Client/Server architecture for Multimedia Database - **Oracle Multimedia Database Management System (OMMDBMS)**. This Client/Server application will enable the user to retrieve, edit, add or delete data in the database in a remote manner that can be done by login on as a "Client" in this application. To have better understanding about Client/Server Architecture and its advantages compare to others architecture such as Centralized Multi-user Architecture and Distributed Single-User Architecture.

1.3 Advantages of Client/Server Database

Now that we have understand the concepts of Client/Server Architecture and Client/Server Database, it is also important to know the advantages of implementing this architecture so that we shall appreciate this architecture much. In this report, I will separate these advantages into 4 categories and then explained these advantages one at a time so that we can have a clear view of each advantage.

1.3.1 Flexibility

The Client/Server architectures is said to be flexible because the application's processes is distributed across many different platforms, with only limited and well-defined server processing being performed on the "shared" resources. This factor results in a performance curve that is more linear throughout and beyond the individual server platform's design capacity (Figure 1.4). Because of this, the cost per user is more predictable and increases in a linear fashion. Furthermore, because of the inherent scalability of the architecture, capacity can be added in much smaller and therefore less costly, increments.

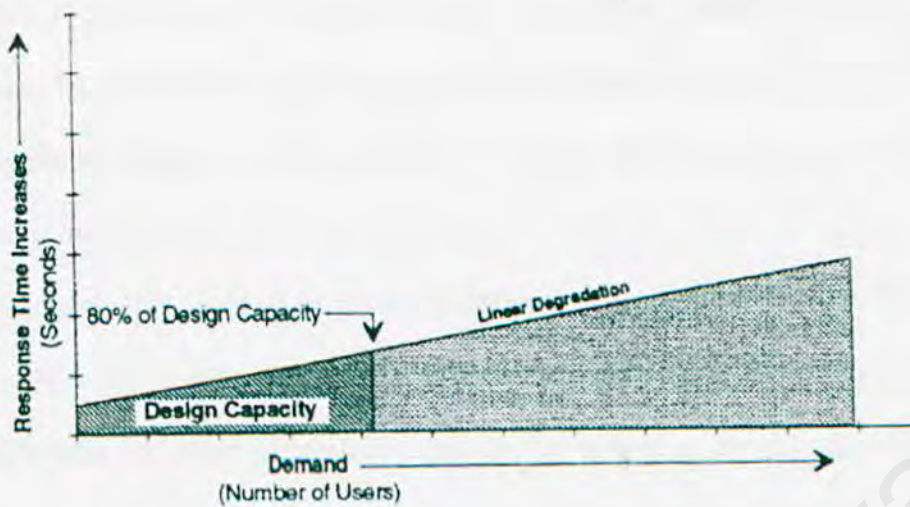


Figure 1.4 : Client/Server architecture performance characteristics.

1.3.2 Scalability

Scalability is a characteristic that describes how easily a given system solution can be scaled upward to maintain acceptable response as demand increases. Closely associated with the performance characteristics of the foundation technologies, this characteristic also factors in the availability of the alternatives for increasing capacity. This is another area where client/server architectures, based on open-industry standard technologies are superior to the proprietary technologies of a multi-user architecture. In a proprietary multi-user environment the customer's upgrade alternatives are limited to what the **original equipment manufacturers (OEM)** or in some case, a **plug-in compatible manufacturer (PCM)** provides. Quite often, in a captive market, the decisions of what alternatives will be available are not driven by the customer need or technology but by market and revenue considerations. This result in an often artificial limitation in

the alternatives made available as the OEM/PCM seeks to maintain account control while causing the customer to migrate to larger and larger systems. Because of the proprietary nature of the technology, the customer rarely has any viable alternative.

Contrast this with the open technologies on which client/server architectures are based. In this environment the market rules and the customer has many alternatives available for adding processing capacity in the incremental steps that are more finely tuned and appropriate to the customer's needs and not the OEM's revenue stream. It is easily possible to cost-effectively scale a client/server application's capacity from 5 users to 200 users in small increments that precisely match increasing demand at an appropriate cost.

1.3.3 Technology potential

Multi-user architectural technologies have, since their inception, been proprietary. Because of this, technological advances have been controlled and directed by small number of original equipment manufacturers who experience little, if any, competition in their market segments. Further, the high cost of research, development, and production of proprietary new technology has historically resulted in relatively long product life cycles. Combine these factors with a very real financial need to protect customer technology investments while maintaining revenue streams, and the results are an incremental and evolution increase in processing capacity.

Compare this with the capacity increases experienced by the technology on which the client/server approach is based. The enabling technologies have been driven by intense competition among a large number of OEMs across the entire range of supporting technologies, from central processing unit (CPU) to memory to disk drives and lastly, to a complete systems. Market dynamics and intensive competition have driven shorter and shorter products cycles, while existing and emerging industries-wide standards have relieved customer concerns about protecting technology investments. In combination, these factors have resulted in an explosive growth in processing capacity, product life cycles that measured in months instead of years, and price/performance ratios that proprietary technologies will never be able to approach.

1.3.4 Cost

Once again the market's influence is primary determining factor. With an installed base acting as essentially a captive market, there is little competitive pressure for proprietary OEMs to compete in cost. The primary OEM consideration in a captive market is to calculate a price point marginally below the customer's cost to migrate to another proprietary technology.

But in the client/server market technology OEMs must compete not just on the technological innovation but also the price. Further, much of the technology is standard and differences between OEMs offerings are

relatively small. Because of this, pricing tends to become the dominant means of differentiating product. We have seen this trend accelerate as even such major manufacturers as IBM and Compaq are being forced to drastically lower prices and open new distribution channels in order to regain market share lost to their lower-priced competitors.

1.4 Scope of Project

We have identified four tasks here in our project. Here are the tasks:

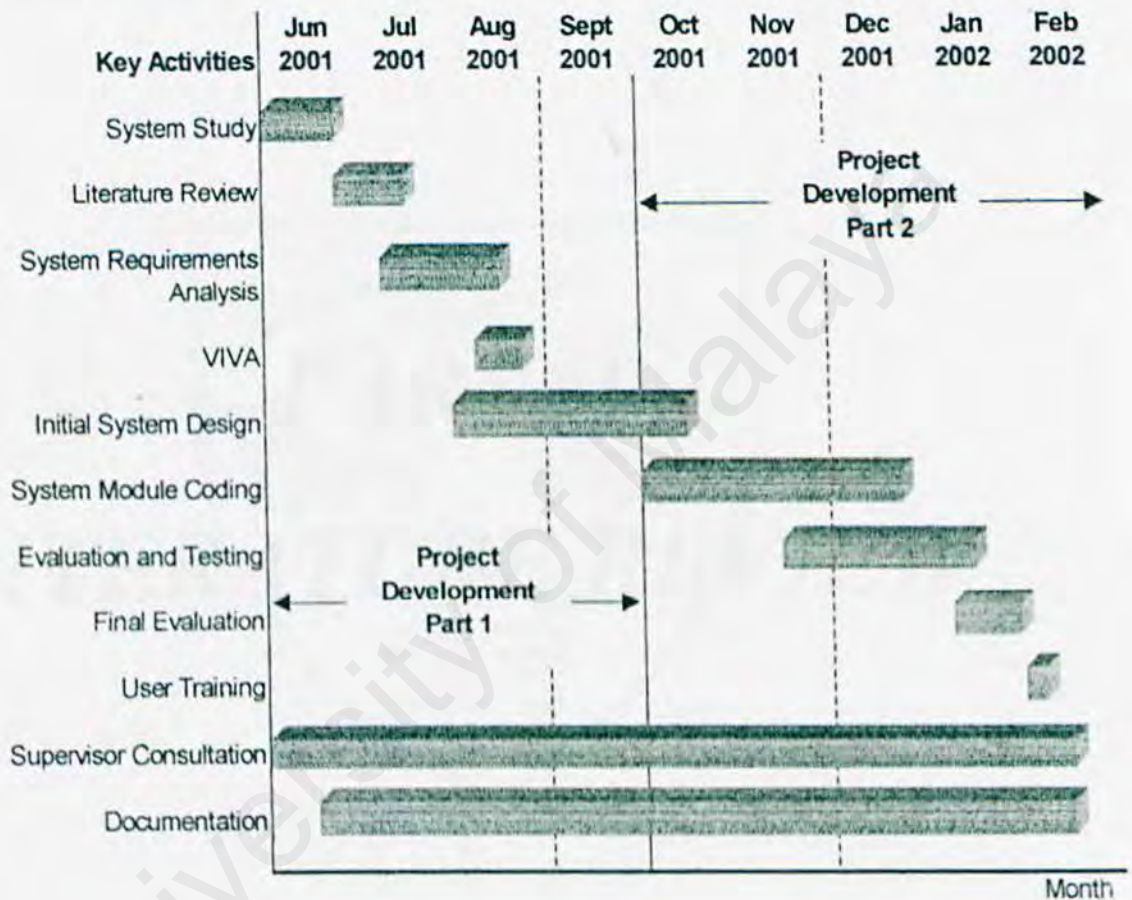
- **Client/Server System Design** – In this task, a client/server connection will be established between the database engine and the remote access computers in secured manner.
- **Client Application** – A window-based application will be built to retrieve data from the data center. This application will consist of a few modules such as data entering and data retrieving modules. This application will be written in Visual Basic 6.

My module, Client/Server System Design and Implementation is responsible of establishing and configuring Client/Server connection plus the task of creating the system's application. These two tasks of mine can be separated to a few more sub-tasks listed below:

- Establishing and configuring a client/server connection between the database engine (server) with the client IBM computers on MS-Windows platform. To establish the connection between the client and the server, the Oracle's SQL*Net/ Net8 middleware will be used.

- Design and create a windows-based query form to enable the user to key in their requests.

1.5 Gantt Chart



Chapter 2 Client Architecture

In this chapter, I will focus on the components of the client. As illustrated in Figure 2.1, the components of the client workstation are basic workstation hardware, the operating system, database connectivity software, applications and graphical-user interface.

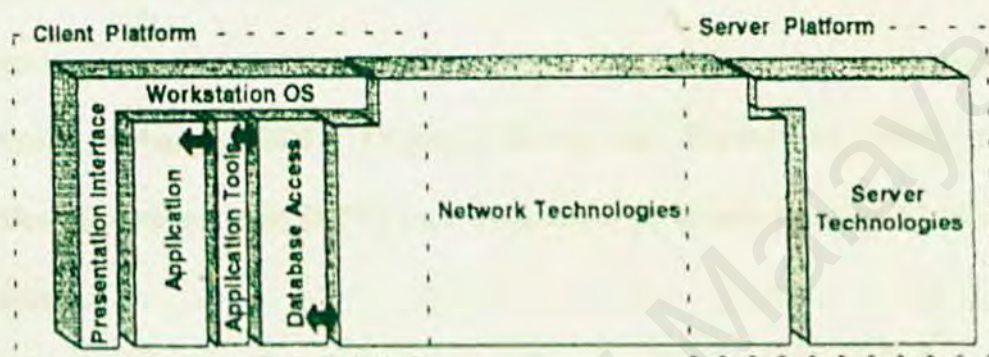


Figure 2.1 : Client Components

2.1 Application and Tools

An aspect that is critically important to the developer's viewpoint is the availability of application development tools. These tools include the various computer languages, text editors, debuggers, design aids and all the other miscellaneous tools that make it possible to design, test, deliver a client/server application.

Client/server developers must be aware of all the various tools and applications, whether purchased or developed internally, already existing in

the client workstation and attempt to leverage the functionality represented by those as much as possible. For example, if there is a requirement for statistical analysis and graphical display, it makes little sense for the developer to include this function in his/her application, if there is already an existing tool (e.g., spreadsheet) that can be incorporated into the application. With the new developments in operating systems and emerging standards in such program-to-program communication mechanisms such as **Dynamic Data Exchange (DDE)**, **Object Linking and Embedding (OLE)** and **Remote-process-calls (RPC)** such integration can appear seamless to the end user.

2.1.1 End-User Tools

Traditionally, applications and tools were generally viewed as falling into clearly delineated categories. The first were such basic productivity applications such as word processing, spreadsheet and presentation tools. These were used by the end-user to perform basic administrative and office functions. The second category included those easily used semi-programmable tools that facilitated data query and reporting activities. Once again these were considered as primarily end-user tools. Then there were applications that have come to be considered as workgroup applications, or those that facilitated communication among end users, with electronic mail

being the most prevalent example. Finally, there were the “real” development tools, used only by professional application developers. These were the full-featured computer languages, compilers, text editors, debuggers and other miscellaneous tools used by the developer to build production applications.

The distinctions between these categories are rapidly disappearing. With the incorporation of increasingly function macro languages and more easily used ways to link applications together, products that were previously dismissed as “productivity” aids are approaching the capabilities needed by professional application development. Professional developers still view such tools as toys should take a walk around their organizations and see what non-data-processing personnel are doing with their word-processing or spreadsheet macros.

2.1.2 Developer tools

Although the line between professional development tools and traditional productivity applications is blurring, there are still a number of characteristics that serve to distinguish the suitability of a tool for professional development use, especially in the development of what are commonly thought of as line-of-business or mission-critical applications.

- A professional development tool must be able to support large multiperson development efforts by facilitating the sharing and

integration of code developed by many developers across one or more projects. Development tools that are totally self-contained and assume “one developer – one application”, no matter how capable, are too limited in any sizable project.

- A professional development tool must be able to encapsulate its code into some inaccessible and non-modifiable form for distribution to end-users. This encapsulation can take the form of compilation and linkage into binary executables, compression into forms executable only through run-time libraries, or any other mechanism that serves to protect the application from unauthorized modification.
- A professional development tool, especially in client/server environment, must be able to access in as direct a manner as possible, many different types of database servers. Development tools that are tied to a single database management package, are in the long run, too limited in today's multiple database environment.
- A professional development tool must be able to take full advantage of the features and facilities made available to it by the environment in which it operates. If the tool is meant to operate within an environment offering a graphical-user interface, it must take full advantage of that interface in standard ways. If program-to-program communication

facilities are provided (DDE, OLE, RPC, etc), the tools must be able to utilize those facilities.

Any development tools that meet the above constraints should, I believe, be considered a professional development tools.

2.2 Operating System

The primary purpose of the workstation operating system is to provide applications operating on that workstation with access to hardware resources (memory, disk data storage, video and printer output, etc) of that workstation and manages the interfaces between that workstation and devices external to that workstation. Today workstation operating systems are distinguished by four primary technical considerations.

2.3 Graphical-user Interface (GUI) Facilities

This refers to the ability of the operating system to support, include or provide the application(s) executing within that operating system with standard application programming interfaces that allow the users to interact with both their applications and operating system itself in a more intuitive and easily used graphical manner that is consistent across all applications residing in that environment. Although client/server applications do not, in and of themselves have to use GUI to communicate with the end user, the

trend in user interfaces is definitely toward some form of GUI, and client/server applications will almost always utilize such an interface if it is available.

A GUI has two faces and is perceived differently by the end user and application developer. The end user perceives the GUI as the “look and feel” of the interface to the system. Meanwhile the developer perceives the GUI as: Firstly, an **application programming interface (API)** to which the application issues commands to present information to the user and as a mechanism to accept commands initiated by the user and secondly, a set of guidelines that should be followed in order to ensure that the developer’s application will interact with the user in a manner that is consistent with all other applications operating through that GUI.

A GUI can be embedded into, and considered a part of, the operating system itself or provided as extensions to the operating system.

2.4 Hardware Platform

This is the basic client technology foundation on which everything else rests. The hardware is what most of us think of as the “computer system”. The basic hardware components of the client include the central processing unit (CPU), random-access memory (RAM), direct-access storage devices (DASD, also known as disk drives), one or more input devices (keyboard,

mouse, etc) and a video monitor. Client workstations are available in proprietary, semi proprietary, and totally open-industry standard packages to meet every known customer need.

Much has been written about the different design approaches taken to the technologies that make up the hardware of the client workstation. For the purposes of providing a suitable client/server foundation, we have to be concerned about the functions that the platform can provide to us, regardless to how the platform is technically built.

2.5 Database Access

This component provides the application with an interface to which database access requests, most often formatted as structure query languages (SQL) statements, are submitted and verified and through which server responses are returned. SQL as the only recognized standard (or semi standard) mechanism for accessing data across variety of database engines, is the current lingua franca of client/server database applications, and its emergence as a standard is in large part responsible for the popularity and growth in client/server computing.

At its most basic, the database access component consists of two parts as illustrated in Figure 2.2. The first is the component to which the application passes request and from which it receives its data. The second component is

responsible for preparing the verified and formatted SQL statement with any inter-processes communication protocol-specific statements, such as OS/2's **Named Pipes** or **Unix RPC calls**, that may be required to establish or maintain linkage with the database, and then formatting the resulting request in the manner expected by the network's transport protocol. There are two basic approaches to providing the SQL API portion of this access. These two approaches are **Embedded SQL** and **Function call interface**.

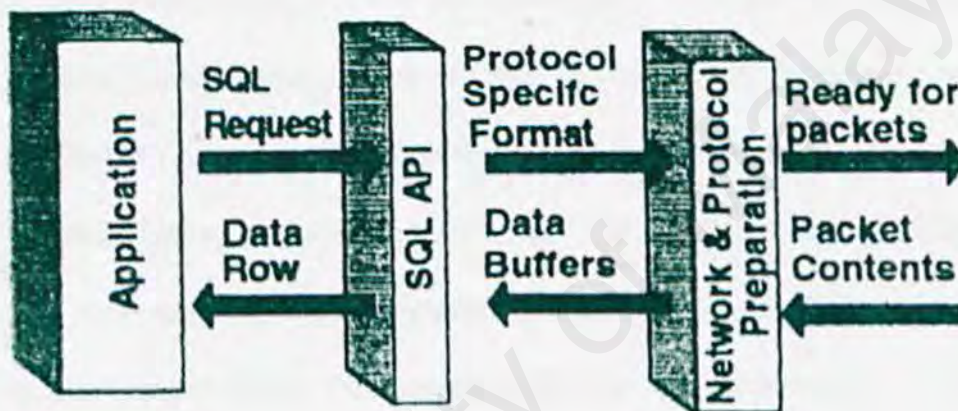


Figure 2.2 : Database connectivity

2.5.1 Embedded SQL

At the most basic level almost all databases support two different approaches to providing this access capability. The first is usually referred to as *embedded SQL* because the programmer embeds the SQL code directly into the program as normal program statements delineated by **EXEC SQL** and **END-SQL** statements. The syntax for this is common across most precompilers offered by the database vendor convention and is based on

IBM's DB/2 conventions, which act as a sort of *de facto* standard. In this approach the programmer writes the SQL code directly into the program in line with the program's other code, with the SQL code delineated by some identifying statements that defined the starting and ending points of the SQL code block. The resulting program is then scanned by software called a precompiler provided by the database vendor. A precompiler is a specialized software which scans the source code being input into the compiler and replaces certain statements with their equivalent in-line assembly language or function call statements, which can then be compiled by the actual compiler and linked into executable code specific to the environment for which the program is being prepared. In the case of a SQL precompiler the embedded SQL statements are most generally replaced with function library provided by the database vendor. This approach offers several advantages.

Firstly, is the relative simplicity and straightforwardness of the program's interface with the database.

Secondly, is the relative portability of the resulting program. Because the majority of precompilers use a common *de facto* standard syntax and most database vendors comply with at least level 1 of the ANSI SQL standard, a little care on the developer's part can result in an application that merely needs to be "recompiled" using a different compiler and database management systems.

2.5.2 Function call interface

The second approach involves directly accessing the function library, again provided by the database vendor, with function calls through which the application passes SQL statements and then retrieving the results of the request through additional function calls which act to bind the data into the specified program variables.

This approach results in more complex programs because the program is handling all the lower-level details previously handled by the precompiler. Another disadvantage to this approach is that function libraries, without either de facto or formal standard to guide them, tend to be highly unique and specific to database management systems. This approach does offer some advantages. The first is that it can be used by any programming language or development tool that can initiate a function call, which is practically everything from desktop application macro languages to C. The second advantage is that, with direct access to the function library, the application has greater flexibility in the type and nature of information it can retrieve about the database itself. The last advantage can be performance. Depending on the efficiency and “smarts” of the supplied precompiler, it is possible for the programmer to produce more efficient database access code by directly accessing the function library.

2.6 *Middleware*

The term middleware applies to a software layer that provides a programming abstraction as well as masking the heterogeneity of the underlying networks, hardware and programming languages [1]. Some middleware such as Java RMI supports only a single programming language. Most middleware is implemented over the Internet protocols. Which themselves mask the differences of the underlying networks.

In addition to solving the problems of heterogeneity, middleware provides a uniform computational model for use by the programmers of the servers and distributed applications. Possible model include remote object invocation, remote event notification, remote SQL access and distributed transaction processing. For example, CORBA provides remote object invocation, which allows an object in a program running on one computer to invoke a method of an object in a program running on another computer. Its implementation hides the fact that messages are passed over a network in order to send the invocation request and its reply.

2.7 *Interprocess Communication Protocols*

As SQL is common language that allows applications to be isolated from but communicate with various relational database products, interprocess communications protocols (IPC) are common language that allows any two

programs running in the same or different environment to send and receive messages, commands and responses. The nature of client/server architecture requires a highly developed interprocess communication protocol to control, synchronize and facilitate the message flow between client and server applications. IPC protocols play a critically important part in client/server architecture and generally responsible for:

- Coordinating a transaction session between a “client” process and a “server” process.
- Pacing the transfer of data between the two processes so that each can complete processing of “old” data before “new” data arrives.
- Making the network location of each process “transparent” to the other process.

Named pipes, remote-procedure calls (RPCs) and application program-to-program communication (APPC) are three most commonly utilized IPC protocols in today’s client/server environments.

2.7.1 Named Pipes

Named pipes is the native IPC protocol of OS/2 and is a fully implemented API that provides interprocess communication between programs in a manner that is very similar to writing to a file, except this “file” is referred to as a “named pipe” that can be shared by many different processes at once. Pipes

can be unidirectional (write or read) or bi-directional (write and read), blocking or nonblocking and dedicated to a single process or used by many processes. An OS/2-specific protocol, named pipes is used extensively by Microsoft's version of SQLServer and until recently, was the only method through which clients could communicate with an OS/2 SQLServer server.

2.7.2 Remote-procedure calls

RPCs are the interprocess communication protocol primarily used in UNIX environments to facilitate program-to-program communication. Although loosely specified by the X-Open standard, RPC protocols have, in the past, been largely proprietary to specific vendors' Unix implementation. With the ongoing consolidation of the Unix world and closer cooperation between Unix International and the Open Systems Foundation, the various forms of vendor-specific RPCs are becoming more standardized. RPCs are implemented as a compiler feature supported by platform-specific RPC run-time libraries that allows the program to initiate remote-procedure calls in the same manner as a local procedure call would be made with the run-time library responsible for "finding" the remote procedure, establishing the transaction session, and handling all the communication, completely transparent to the application. Primarily Unix protocols, RPCs are used

extensively by Unix-based SQL database products to facilitate the connection between client process (application) and server process (database).

2.7.3 Application Program-to-Program Communication (APPC)

APPC is IBM's System Network Architecture (SNA) interprocess communication protocol to facilitate conversational communication between logical units (software) executing on physical units (hardware) across a SNA-compliant network. Developed to support interprocess communication across a wide range of heterogeneous IBM host systems operating within an SNA environment, APPC is a functionally rich and powerful protocol that addresses security, remote-program initiation, distributed checkpoints and synchronized half-duplex. APPC is supported by IBM Common Programming Interface for Communications (CPI-C) software provided for each IBM-supported platform (e.g. DOS, OS/2, OS/400, VM, MVS, etc.). In the client/server architecture APPC is the IPC protocol of choice for client to such IBM database management products as OS/2 Data Manager, SQL/400, SQL/DS and DB/2.

Chapter 3 Network Technologies

An explicit characteristic of the client/server architecture is that there must be a communication linkage between the client and server platforms across which the client's request for data and the server's response are communicated. This communication linkage can take many different forms, from dial-up access over switched public phone lines to the typical dedicated point-to-point SNA networks found in most large IBM mainframe environments. While any reliable and fast linkage can support a client/server application, such linkages tend to be highly specific to the requirements of a given environment, and the more commonly used approach to establishing connectivity between clients and servers involves the use of local area networking technologies.

3.1 Local Area Network (LAN)

A local area network is that collection of networking hardware, cabling and protocols that work together to provide a method for computers and data systems to connect and share cabling. Protocols in the context of local area network are simply rules that define how these various components will work together at each level. Often, these rules are codified into public domain by

public organizations set up specifically to define and oversee the definition and modification of these standards.

The two major approaches to area networking that adhere to standard defined and managed by the IEEE are Ethernet (overseen by the 802.3 subcommittee) and token ring (overseen by the 802.5 subcommittee). Each of these different approaches combine cabling topologies, signaling techniques, information packaging, and access control protocols encompassing the physical and data-link control layers of networking in different ways to meet the objective of being able to pass a packet of information across a local area networking.

At its simplest level a packet can be thought of as an envelope that contains information being passed by the network. This envelope carries a destination address for the recipient and return address for the sender. The ways in which these addresses are specified, the format of the information inside the envelope, and the way the accuracy and completeness of the information are verified are determined by protocols.

3.1.1 Ethernet

The Ethernet was developed at the Xerox Palo Research Center in 1973 as part of the programme of research carried there on personal workstations and distributed systems [2]. The pilot Ethernet was the first high-speed local

network, demonstrating the feasibility and usefulness of high-speed local networks linking computers on a single site, allowing them to communicate at high transmission speeds with low error rates and without switching delays. The original prototype Ethernet ran at 3Mbps. Ethernet systems are now available with bandwidths ranging from 10Mbps to 1000Mbps. Many proprietary networks have been implemented using the same principle basic method of operation with cost/performance characteristics suitable for variety of applications. At the lowest cost level, the same principles of operation are used to connect low-cost microcomputers with transmission speeds of 100-200 Kbps [3].

3.1.1.1 Topology

The standard Ethernet topology is a simple branching bus-like connection line using a transmission medium consisting one or more continuous segments of cable linked by hubs or repeater (Figure 3.1) [4]. This cabling scheme offers the advantages of simplicity and low cost at the cost of reliability and flexibility. In terms of reliability the biggest weakness of a bus topology is the fact that a break anywhere in the bus will bring the network down. In terms of flexibility, a bus, although easier to set up initially, becomes

increasingly more difficult to configure as new devices are added to the network or existing devices are moved to different locations.



Figure 3.1 : Standard Ethernet bus topology

To address some of these problems the 802.3 committee created a standard called the 10BaseT, defining the use of twisted-pair cabling configured physically as a star topology centered around intelligent hub concentrators, illustrated in Figure 3.2. The star topology is uniquely suited to the already existing typical telephone wiring schemes in widespread use and significantly increases the flexibility with which new nodes can be attached and existing nodes moved around.



Figure 3.2 : Typical 10BaseT topology

Although physically a star topology, 10BaseT is still, electrically, a bus and to address the “cable break” weakness of an electrical “bus” the 10BaseT standard calls for an intelligent hub that can detect such break and automatically drop the broken cable segment and reconfigure the bus around the break, thus retaining network integrity. Ethernet is limited, in either of its topology to a maximum of 1000 nodes on a single network segment.

3.1.1.2 CSMA/CD

The second defining characteristic of Ethernet is the protocol by which it allows shared transmission access to the cable. Until such new technologies as asynchronous transmission mode become standardized and in more widespread use only a single device can access any baseband network at a time. Media access protocols are thus necessary to allow multiple devices to share a single medium at once. In Ethernet, this protocol is called CSMA/CD and requires each station to “listen” before it transmits. If the line is “busy”, the station must wait until it isn’t. If two stations happen to listen at the same time, hear nothing and transmit simultaneously, the “collision” results in a scrambled message, which both stations detect. Each station then waits for a predetermined yet random period and tries to transmit again. Think of CSMA/CD as working just like your telephone. Each person can speak only

when the other is listening. If they both happen to speak at once the message is garbled, which both recognize and each person waits to listen before speaking again.

3.1.2 Token Ring

Based on patents awarded to Olof Soderblom of the Netherlands, codified into standards by IEEE 802.5 subcommittee and popularized by IBM as their standard local area network, the token ring network is the new guy on the block, being introduced IBM in the late 80's. The definitive operational characteristics of token ring are its dual speeds (4 and 16 Mbps), star/ring topology and the means it uses to arbitrate access to the transmission medium, token passing.

3.1.2.1 Topology

The only topology used by the token ring networks is a ring, or complete circle, that is implemented physically as a star using either active (smart) or passive (stupid) hubs called media access units (MAUs), as illustrated in Figure 3.3. This cabling scheme offers the same advantages of Ethernet's 10BaseT topology. The reliability of token ring networks is not an issue of the topology, as with Ethernet, but of the media access method used.

Although physically a star topology, token ring is electrically a ring with each node being connected to two other nodes, the one “left” and the one “right”, and the electrical connection is a complete unbroken circle. Token rings MAUs are normally eight port concentrators with two additional ports that are specifically used to connect two MAUs together, thereby extending the “ring” to additional devices. A single “ring” can consist of up to 32 connected MAUs or a maximum of 256 nodes.

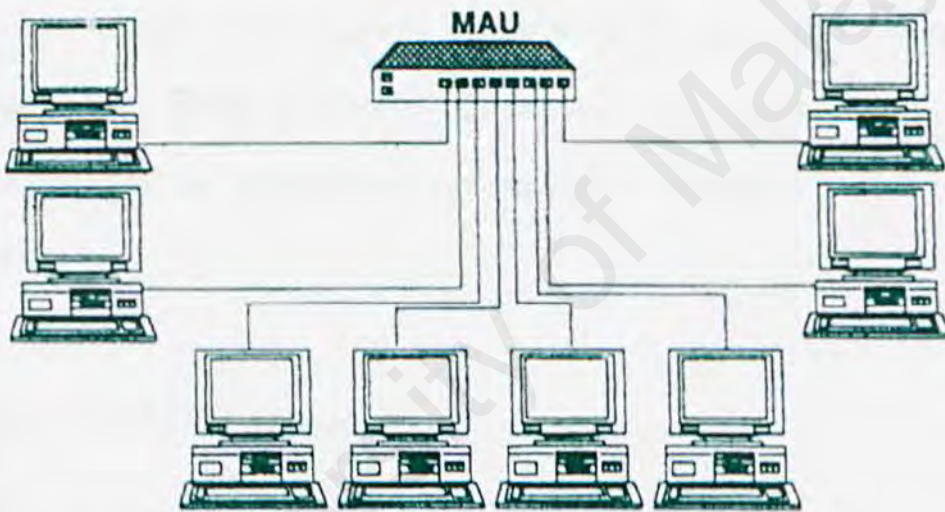


Figure 3.3 : Typical token ring topology

3.1.2.2 Token Passing

The second defining characteristic of token ring is the protocol by which it allows shared transmission access to the cable. In a token ring network a

single electronic “token” is passed around the ring from node to node. If the token is “free” when it reaches a node, that node can attach a message to it, set it to “busy”, and pass it along to the next node. The token is then passed from node to node until it gets to the node for which the message is intended. That node reads the message and passes the token along, still busy. When the token returns to the sender sets the token to “free” and passes it along. Theoretically, with limits on how long any one node can hold a token “busy” and known number of nodes (hence the 256-node limit), it is possible to calculate exactly how long the maximum length of time could be before any given node would be allowed access. A token ring network is thus characterized as “deterministic” as opposed to Ethernet’s “probabilistic” nature.

3.2 The OSI Model

Any discussion of components that go into making up a network, whether Ethernet or token ring, should begin with a basic overview of the International Standards Organization’s (ISO) Open Systems Interconnect (OSI) model. The OSI model grew out of a recognized need to establish a standard, nonproprietary, network architecture that could be a stable

foundation on which to build a network that would interconnect proprietary computers from many different OEM sources.

A protocol, in the context of computer communication can be best thought as simply a set of rules that hardware and software components use to describe and govern the valid types of messages that can be sent between two parties. The key elements of a protocol are its syntax, semantics and for electrical signal protocols, timing specifications. The syntax of a protocol essentially describes the standard format of a message. Protocol syntax describes the allowable contents of a message in terms of how it is structured. Timing, a set of rules primarily associated with electrical signals, defines the mechanisms by which messages that may travel at different speeds and arrive in a different order than they were sent can be sorted out and processed.

As illustrated in Figure 3.4, the OSI model is a partition of system's functions into seven distinct layers. Each layer describes one aspect of the system environment and the protocol that govern that layer's interaction with the layers directly above or below it.

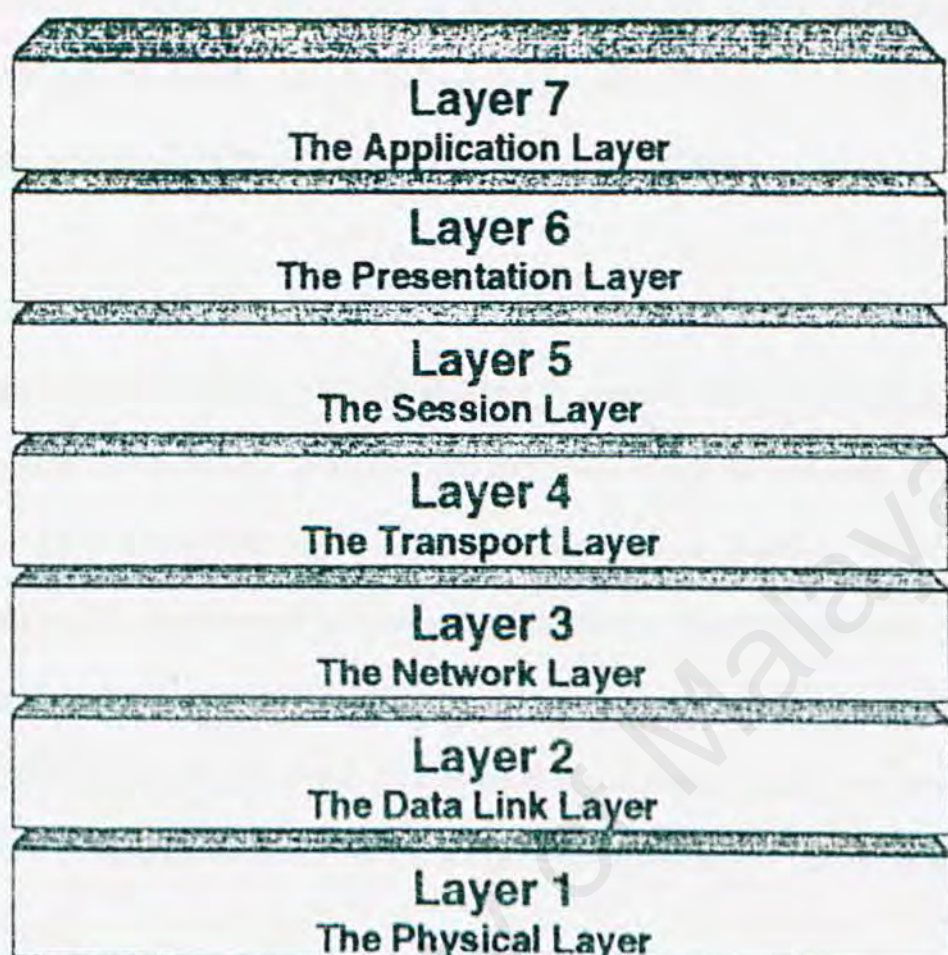


Figure 3.4 : OSI architecture

3.2.1 Layer 1: Physical layer

The physical layer of the OSI model deals with the hardware required to provide a connection between two devices. The components most often thought of as residing at the physical layer are cabling through which the signal moves, repeaters that are sometimes used to extend the allowable

distance between two nodes, any intercable connections such as patch panels or punchdown blocks that may be used, and it, in some cases, any centralized concentrators or hubs used to establish star configurations.

3.2.2 Layer 2: Data-link layer

Once a physical connection is established, the signals through that connection need to be interpreted, generated and controlled in accordance with a set of protocols. At this level of the model these protocols are primarily electrical in nature and implemented in hardware assisted by software to move the data represented by signal from the wire to computer memory where it can be further processed. In a local area network the network adapter card and the network card driver are the primary examples of components at the data-link layer.

3.2.3 Layer 3: Network Layer

The network layer is concerned primarily with determining the physical routing of information across the network and is of particular importance in wide area networks where intelligent routing of packets between nodes that may be accessible from many different paths is very important. This layer normally does not play a significant part in typical local area network, as all

the node addresses are "local" and generally accessible from only a single path.

3.2.4 Layer 4: Transport Layer

The transport layer performs the addressing and routing functions for nodes that are "local" to the network. The purpose of this layer is the packaging and logical addressing of outgoing information and the unpackaging and reading of the contents of arriving packets. It is at this layer that the various types of LAN protocols and network operating systems begin to manifest their own uniqueness. Novell's Netware, for example, uses IPX to implement this layer, while Unix systems generally base their communication on TCPAP. It is at layer 4 and above that today's LAN protocols and components begin to diverge from the OSI model, combining many of the functions of layers 4, 5, and sometimes 6 into a single protocol or particular piece of software. The term, "protocol stack" is often used to describe components such as IPX, NetBios, and TCP/IP that reside at this layer.

3.2.5 Layer 5: Session Control Layer

This layer is responsible for implementing the protocols that allow two programs to establish and control a session of communication. A session can

be thought of as a group of transactions or a logical unit of work that can be marked by a beginning, n transactions of a specific type (security, administration, data request, remote program execution, etc.), and an end. The functions this layer performs are critically important to client/server computing and are often embedded, along with transport layer functions, into the same software. Novell's session control protocol, SPX, is implemented along with IPX in the same software, while Microsoft's Named Pipes and the server message block (SMB) protocol used by NetBios perform functions at this level.

3.2.6 Layer 6: Presentation layer

The top two layers of the OSI model deal less with networking functions, as I think of them, but with operating system and application interfaces, and, in my opinion, the higher you go in the OSI model, the fuzzier the distinctions get between what function goes into what layer.

The OSI presentation layer is responsible for providing the interfaces between the application and the various services and resources required by that application. The difficulty lies in defining "application." For example, the OSI model would characterize certain parts of operating systems, GUIs, and database management software as "applications" in the same sense that

an order entry system or word-processing software are applications. This makes the boundary between layers 6 and 7 somewhat indistinct, leaving only such low-level functions as the basic input/output system (BIOS), video drivers, and disk drivers as clearly layer 6 functions. Arguments could be made for components such as Microsoft's MAPI (mail application programming interface), GUI APIs, the X-Window's protocols, Unix's "sockets," and possibly database access APIs for categorization as residing in either layer 6 or layer 7.

3.2.7 Layer 7: Application layer

The top layer is the application itself or that portion with which the user interacts directly. It is responsible for supporting network applications. The application layer includes many protocols, including HTTP to support the Web, SMTP to support electronic mail, and FTP to support file transfer [5].

3.3 Node Components

These are the network components that are installed on both the client and server nodes that provide physical connection to the network, apply the appropriate protocols to govern normal network specific communication between all layers, and, in the case of a database server, provide the network-

specific layer that allows the database access functions to communicate with the appropriate network protocols.

3.3.1 DB-net interface

With most, not necessarily all, database servers the database access functions contain a lower-level layer that isolates the database's data access layer, the call-level function library, from the intricacies of the network-specific transport/session level protocols and that may add additional command and unique session control facilities of its own. Examples of this component include Microsoft's DBMSSPX3.DLL implementing IPX/SPX protocols for SQLServer, Novell's OS/2 requester providing named pipes support for Microsoft's OS/2 SQLServer on Novell networks, and Sybase's series of Netlib facilities that provide support for Sybase's SQLServer operating under a number of different operating systems to be accessed across different transport protocols. Another example of this category is Gupta's SQLRouter, which provides a single development tool or family of tools, in this case Gupta's SQLWindows, Quest, etc., with access to many different database servers across many different transport protocols. These classes of components exist at level 6 or level 7 of the OSI model.



Figure 3.5 : DB-net interface in Client/Server model

3.3.2 Network protocol

The network protocol software in today's LAN software is responsible for the OSI level 4 and/or level 5 functions of logically addressing a packet, correctly formatting the information put into the packet, sometimes implementing session control facilities, and passing the resulting packet directly to the network adapter driver at the data-link level. This component is also responsible for accepting the hand-off of incoming packets from the network adapter driver, verifying the format and accuracy of their contents, and passing the resulting data to either the session layer or directly to the presentation layer depending on configuration and network. Examples of components that reside at this level include **IPX/SPX**, **NetBios**, and **TCP/IP**.

3.3.3 Network adapter driver

The driver is a small piece of software, generally written by the adapter's manufacturer that is specific to the network card and controls its operation and how it moves data to and from the computer's memory. The major factor in the performance of a network adapter card is often the efficiency of this driver. For instance, 8- and 16-bit network adapter cards from one vendor can often outperform superior 32-bit bus mastering cards from other vendors simply on the basis of more efficient driver software. A de facto standard method of interfacing with network adapter drivers grew up over the years and, as the vast majority of LANs traditionally used a single transport protocol (i.e., IPX/SPX or TCP/IP), this de facto standard was sufficient to establish interoperability between various network adapter card vendors and network operating system products.

The advent of client/server applications added complexity to this situation by creating the need for a client platform to simultaneously connect to various different servers that "spoke" different transport protocols. One of the alternatives developed to meet this need is to make it possible for different transport protocol software to "share" access to the network adapter driver, something that was impossible under the original de facto standard. To facilitate these two different standards for drivers, Microsoft's Network

Device Interface Standard (NDIS) and Novell's Open Datalink Interface (ODI) were developed. Drivers written to one or the other (or both) allow those network adapter cards to simultaneously support multiple different protocols through the same card. If you contemplate needing such a capability, then compliance with NDIS, ODI, or both should be a consideration in the selection of network adapters.

3.3.4 Network adapter card

The network adapter card physically connects the client and server to the network medium, accepts, encodes and/or decodes packets, and sends packets on the cable in accordance with the transmission and media access protocols (i.e., Ethernet or token ring) of that network. One of the main functions of the network adapter is to be able to read packet addresses and determine whether the packet is meant for the adapter's node and, if it is, to decode its contents. The ways, in which these addresses are specified, the format of the information inside the envelope, and the way the accuracy and completeness of the information are verified is determined by the network's protocol. Network adapter cards are specific to both the network and the type of cabling being used. For example, there are Ethernet coax (coaxial cable), thin coax, and 10BaseT cards.

Two primary hardware characteristics play a role in the performance of a network card, the width of the bus interface and the mechanism used to move data from the card to computer memory. Adapter cards are available in 8-, 16-, and 32-bit versions, and the wider the data path, the faster the card can transfer data to and from the computer itself. The second factor is the method used by the card to pass "data" to the computer itself. Programmed I/O, direct memory access (DMA), shared memory, and bus mastering are various techniques used; shared memory and bus mastering are the fastest and most efficient methods. For client purposes the method used is relatively insignificant as the throughput is rarely high enough to make a noticeable difference one way or another. This is somewhat more important in adapter cards meant for servers. By their nature, servers are shared resources that can normally expect a much higher throughput than can a client with a corresponding greater need for maximum performance.

3.4 Network Connectivity Components

These include the network components that exist external to the client and server platforms and provide the physical connection between the network adapter cards installed on the client and server nodes. In the simplest possible two-node Ethernet network this could be limited to a single piece of wire. In

complex networks it can consist of many different components, each providing a different piece of the connectivity puzzle. The most common of the components found in this group are described in the following paragraphs.

3.5 Media

The media through which the signal passes are most often a cable that establishes a physical connection between devices on the network. These media are considered to exist at level 1 of the OSI model. The three basic types of cable used are coaxial cable, similar to what you find in your house for cable TV; twisted-pair wiring, similar to that used in telephone installations; and fiber-optic cabling. Each of these types has advantages and disadvantages, differing performance characteristics, and varies widely in cost. Further, there are subtypes of each specially created to meet various operating conditions and requirements.

3.5.1 Coaxial Cable

Coax cable consists of a central copper core surrounded by substantial metallic shielding, one or more layers of insulation, and wrapped in a thick plastic or rubber casing. It is supplied in a number of different types that are

determined by its design and the materials used in its construction. These types are identified by ratings of the cable's impedance characteristics. These ratings are important because different network cabling schemes require cables with specific characteristics. Coax, in a variety of different ratings, is manufactured in both thick and thin versions, and its primary advantages are its ability to carry a signal for a relatively long distance (1000 to 1500 ft) without needing to have its signal boosted and its resistance to external electromagnetic interference, and it is often used in environments where such interference is extensive. Its primary disadvantage is that it is difficult to install, particularly in small cable runs where room is limited, because of its lack of flexibility and its size.

3.5.2 Twisted-pair Cable

Twisted-pair cabling is manufactured in a number of different configurations and exists in both shielded and unshielded versions. Unshielded twisted-pair cable is the least expensive of the cabling options and is rapidly gaining in popularity because of its low cost and the ease with which it can be installed. Unshielded twisted pair is the foundation of most building telephone systems, and it is often possible to utilize unused telephone pairs in newer buildings, although care should be taken to ensure that such pairs are tested and of data-

grade quality. Twisted-pair cabling also comes in a variety of different shielded versions that can approach the shielding characteristics of coaxial cable. The primary advantages of twisted-pair cable, in either shielded or unshielded versions, are its low cost and ease of installation. The primary disadvantage is the relatively limited run length (i.e., distance between two points) of 250 to 350 ft and, in unshielded versions, its lack of resistance to external electromagnetic interference.

3.5.3 Fiber-optic cable

Fiber-optic cable is cable where the normal copper wire has been replaced by glass fibers through which light is used to pass signals, not electric current. This is the most expensive of the cabling options in terms of both the cost of the cable itself and, because of the special knowledge and tools needed, its installation cost. A popular misconception is that speed is the primary advantage of fiber-optic cable. In fact, electric signals move through copper practically as fast as light travels through glass fibers. The main advantages of fiber-optic cable are primarily transmission distance and reliability. Fiber-optic cable can easily transmit signals up to 2 miles without boosting. Further, fiber-optic signals cannot be disrupted by external electromagnetic interference.

3.5.4 Repeaters

A repeater is basically a simple protocol independent signal booster that is used wherever a single point-to-point run of cable exceeds the distance rating of the cabling being used. A repeater intercepts the signal, retimes it, and then retransmits the signal as it was when transmitted from its original source. Repeaters are considered OSI level 1 components.

3.5.5 Hubs

The simplest way to interconnect LANs is to use a hub. Hub is a simple device takes an input (that is, a frame bits) and retransmits the input on the hub's outgoing ports. Hubs are essentially repeaters, operating on bits. They are thus physical-layer devices. When a bit comes into the hub interface, the hub simply broadcasts the bit on all the other interfaces [6].

3.5.6 Concentrators.

LANs that implement a star topology (token ring, Ethernet 10BaseT, etc.) require one or more semicentralized hubs to which media runs from the nodes. These hubs are variously known as multistation access units or concentrators, are generally protocol-specific, and provide centralized point-to-point connections for each node to the hub. These hubs can be

interconnected with other hubs, allowing the network to grow to its specified maximum number of nodes. These hubs can range from the simple and inexpensive standard passive IBM eight-port token ring MAU to complex, multifunction, rackmounted concentrators such as the Synoptics 3000 series, which can themselves host multiple MAUs of different protocols in internal expansion slots. In the typical star topology the cabling does not run directly from the node to the concentrator but to an intermediate wire-to-wire component that simplifies maintenance and reconfiguration of network connections. Punchdown blocks, AT&T type 100 wirewrap boards, and RJ-45 patch panels are examples of the components most often used to provide this wire-to-wire connection. Wire-to-wire components and basic concentrators are essentially OSI level 1 components. Larger, more complex concentrator technologies can, depending on their flexibility, exist and host other components that exist throughout levels 1, 2, and 3 of the OSI model.

3.5.7 Bridge

A bridge is a hardware component that serves to connect two separate networks, passing packets across from one to the other only when the packet is addressed to a node on the other network. Only one bridge is required to interconnect two physically adjacent networks, but two bridges are required

when the networks are separated by distances greater than the node-to-node restrictions applicable to the network. Bridges are often used to physically segment large local area networks into smaller segments in order to control throughput volume, increase performance, link two networks using different types of cabling, or link two networks using different MAC protocols (i.e., token ring to Ethernet). Bridges maintain addresses of the nodes on each side of the bridge and only pass packets from one side that are addressed to the other. Bridges operate at the data link layer or level 2 of the OSI model.

3.5.8 Router and Brouter

A router is a hardware component that is similar to a bridge but which is generally used to serve wider geographic areas and serve to link any two points by encoding special routing instructions that can be used by the third network that serves to connect the two separate routers to intelligently select different paths for the transmission. Routers can server to interconnect networks that use completely different MAC and transport layer protocols. Routers do not maintain node addresses for the interconnected networks and must be addressed specifically by the nodes or other routers seeking transmission of their packets. The functions of the router place it at level 3 of the OSI model.

A brouter is a hardware component that combines the functions of bridges and routers.

3.5.9 Gateway

Consisting of both hardware and software, this component allows a network to be "attached" to another significantly different network using different MAC and transport protocols. From the perspective of the "external" network, the gateway appears as a device standard to that network. Novell's Netware SNA is an example of a commonly found gateway. It consists of hardware (expansion card) that connects to an IBM SNA network and software that makes that gateway "look like" a 327x terminal control unit to the mainframe and devices on that network "look like" attached 3279 terminal devices

Chapter 4 Server Architecture

The server functions as the shared resource half of the client/server equation. As such it must be able to service multiple often simultaneous, requests for data from the various application clients throughout the network. The technology of server platform can range from basically equivalent to the client up to and including large mainframes. This server flexibility is the foundation of client/server architecture's scalability one of the two primary shared resources of the architecture (cable bandwidth is the other), the server provides the processor cycles, data storage subsystems, memory, and bus bandwidth shared by hundreds of clients, and relatively minor upgrades to the server's capacity can have immediate and disproportionately beneficial effects on the perceived performance of the client application.

The downside of this characteristic is that, just like any multiuser mainframe system, its failure drops not just a single user of the application but all the users of all the applications depending on it for service. Thus, the driving factors in selection configuration of server technologies are stability first, performance second, and flexibility third. While it is rarely advantageous to base client platforms on proprietary technologies, proprietary technologies can often provide substantial benefits in stability and performance, provided connectivity can be established.

Another factor to consider in the selection and configuration of the server technology is to determine what functions it will perform. For example, it is quite possible to set up and operate a client/server environment without a network operating system to manage access to the network's shared resources. Although possible, it is rarely recommended, and if the full value of the network and shared resources are to be realized, a network operating system is essential. But the client/server architecture provides significant flexibility in how that environment, as a whole, is configured. For example, if a network operating system is present, the database services can be based on the same platform as the network operating system, sharing access to processor and data storage with other typical network functions, or the database services can be located on an entirely separate platform dedicated to database services, as illustrated in Figure 4.1 (see also Figure 4.2).

A network can have many different servers, all playing different roles, providing the same or different services. The dedicated database server's primary function is to manage and respond to multiple simultaneous requests to the database management software. File servers, generally the "home" of the network operating system, are multipurpose, providing, at a minimum, network security and shared access to files. In addition, file servers, as the home of the network operating system, also often provide for the sharing of network printers, implement communication gateways to the outside world,

gather network performance statistics, and provide administration and management functions necessary to enhance the environment's stability. In some situations it can be advantageous to distribute some or all of these services across many different servers, each dedicated to a specific task (print server, communication server, etc.).

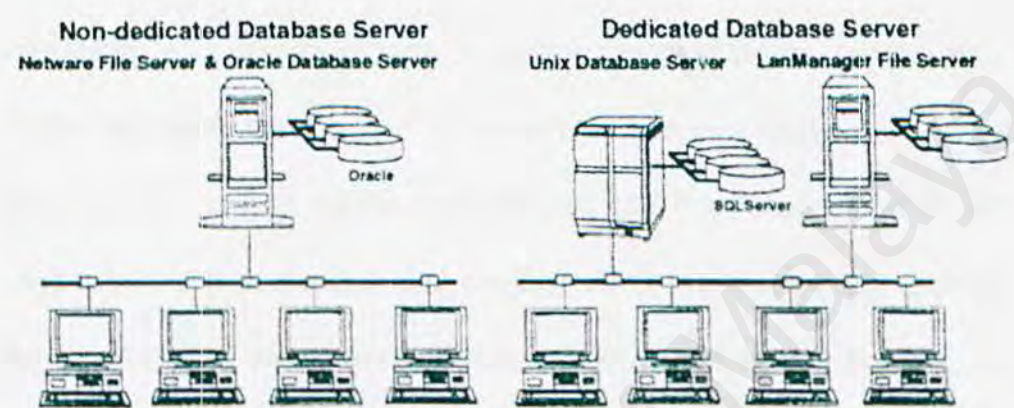


Figure 4.1 : Dedicated versus non-dedicated

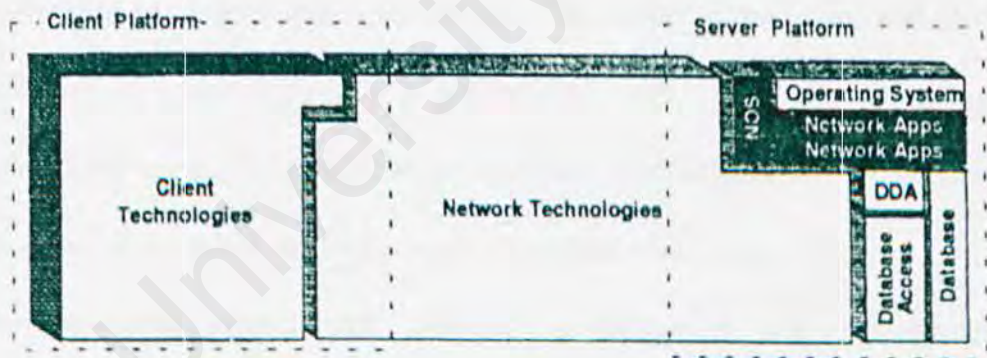


Figure 4.2 : Server operating system

4.1 Server Operating System

Like the client's operating system, the primary purpose of the server operating system is to provide applications operating on that server with access to the hardware resources (memory, disk data storage, video and printer output, etc.) of that server and manage the interfaces between that workstation and devices external to that workstation (video, printer, etc.).

Unlike the client, the nature of the server's mission as a shared resource for many clients requires greater sophistication and functionality beyond that needed by the typical client, and the facilities it provides closely resemble those provided by proprietary-multiuser operating systems. For example, the ability to multitask is not an option but a requirement, and preemptive multitasking, providing greater tolerance for program failure, is generally preferable to cooperative multitasking. The ability to directly address large amounts of memory is also a prerequisite, with 16 Mbytes of storage considered a basic minimum to provide basic functionality and the capacity to extend that storage far beyond this a significant advantage.

An operating system that provides administration and management capabilities, access security, shared client access to files, printer queuing and routing, and various other functions is generally referred to as a network operating system (NOS) and can take a number of different forms. Some network operating systems are full-function operating systems that have been

designed specifically to meet the needs of networked resource sharing. Novell's Netware and Banyan's Vines are examples of this approach. Others, such as Microsoft's LanManager and the network file system (NFS) extensions to Unix, are extensions to multitasking operating systems that add the security, administration and management, routing, and other functions that are specific to networked resource sharing.

Network operating systems also take two different architecture approaches to sharing resources across the network. The peer-to-peer approach blurs the distinction between client and server by essentially making it possible for every network node, running that operating system, to play the role of either client or server. Each such node can be servicing a specific user as workstation while simultaneously sharing its data storage, processor, or printer resources with the network. Unix is an example of such a peer-to-peer approach, while OS/2 workstations in LanManager networks can play a limited peer-to-peer role. The network server approach requires the dedication of one or more specific platforms as the network operating system's "home," and it is only those platforms whose resources can be shared. Whether peer-to-peer or dedicated, a full-featured operating system or simply an extension, the network operating system should provide, at a minimum, the following functions:

- *Resource sharing* - provide controlled access to the basic shared resources of the network.
- *Administration* - the ability to identify and define the authorized users of the network, the resources those users can access, the locations of those users, and other information that aids in controlling and managing access to network resources.
- *Resource management* - those functions that allow diagnosis and correction of problems on both the network and the server, monitoring of resource utilization, the ability to optimize performance of the server in specific situations, and provision of capacity planning capabilities. This area is one where proprietary approaches often offer vastly superior functionality over the more open operating systems and network operating systems available for industry standard hardware configurations.
- *Fault tolerance* - the ability of a server to recover from the failure of one or more of its components without interrupting service to the network. There are many different levels and types of fault tolerance. At the lowest level of fault tolerance is the operating system's ability to survive the abnormal termination of a program executing within that operating system. Higher levels of fault tolerance can include error-correcting memory, disk mirroring (data written to two disks at same time so that if

one breaks, the other takes up the slack automatically), disk duplexing (where an entire disk subsystem including the disk controller is duplexed so that if a controller breaks, the alternative subsystem kicks in automatically), and even server duplexing where an entire duplicate server is standing by being constantly updated so that it takes over instantaneously if the primary server fails. As you might guess, the greater the fault tolerance, the greater the expense. This is another area where proprietary approaches often offer vastly superior functionality. These operating systems and hardware configurations have provided these capabilities for years; while solutions based on industry standard technologies are just beginning to provide the same functionality as "add-ons" to the basic platforms.

4.2 *Server Hardware*

This is the basic server technology foundation on which everything else rests. Server hardware can range from basically equivalent to the client to large proprietary configurations. Unlike the client, the server must be able to provide responsive services to multiple simultaneous client requests while ensuring a stable and secure environment. The disk storage subsystem is possibly the single most important determining factor in a server's performance and reliability. The vast majority of client requests will be for

data, and the disk storage subsystem represent the "choke point" through which these requests must pass. Further, as the major "mechanical" moving part in a configuration, the hard disks that make up the disk storage subsystem are the most prone to failure of any component of the platform.

Given these factors, special attention should be directed toward selecting and configuring the disk storage subsystems of any platform expected to perform a significant server role in one or more applications. Storage subsystems are available in a variety of forms, both proprietary and based on industry standards, each offering their own advantages. Suffice it to say, without going into great technical detail, that a server's disk storage subsystem should provide for very high-speed transfer of data from disk drive to system memory, multiple concurrent paths to and from data so that requests are not single-threaded, and some form of seek optimization so that "data" can be located efficiently and quickly.

4.3 Database Access

This component can either stand alone as an easily recognized module or be embedded in the architecture of the database management system itself, but, in either case, it is specific to the database management software being used and functions essentially as a request manager. This component serves to isolate the database engine itself from the network protocols being used,

manages the client connection to the database engine, translates and transmits the client request to the engine, accepts and sometimes buffers the request result, and interacts with the network protocol to transmit those results back to the client.

This component is often a controlling factor in a database engine's resource usage and sometimes serves to constrain how many clients may be supported simultaneously. For example, early versions of Oracle for the OS/2 platform required 256K (256 kbytes of memory) or more for each client connection due to Oracle's internal architecture. Although this architecture enhanced performance for a limited number of clients, it served as a significant limit to the total number of simultaneous connections that could be supported in OS/2 1.x's 16-Mbyte address spaces. In comparison, SQLServer's internal architecture made effective use of OS/2 threads and reentrancy to reduce resource per connection requirements to less than 60K, therefore trading off some individual client performance under lightly loaded conditions to the ability to adequately support a far larger number of simultaneous connections.

4.4 Distributed Data Access

This function may or may not be present depending on the database software being used. When present, it will be specific to the database management

software and may be provided as a standard facility or as an extra cost option. The purpose of this component, where it is present, is to facilitate and manage communication between two or more database servers residing on similar or different platforms across the network. This communication most generally takes the form of requests for data or updates to data that are "remote" from the server making the requests. For example, Figure 4.3 illustrates a client communicating a request to database server 1. Database server 1 does not have that data available locally but knows where it is and "forwards" the request to the remote server 2, who then responds. Database server 1 then returns the data to the client.

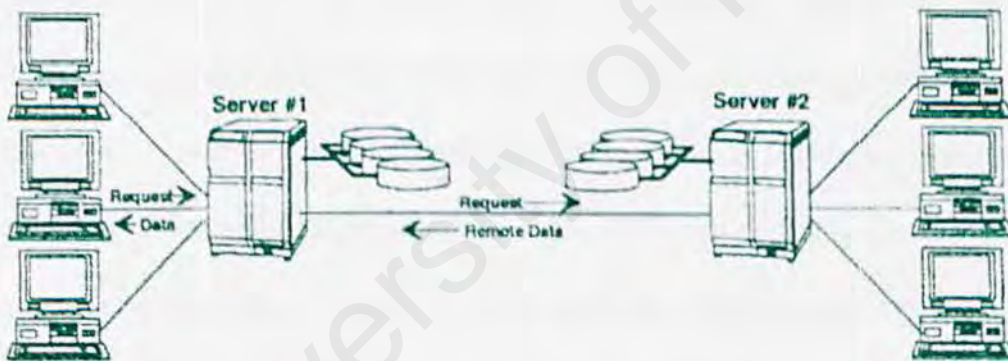


Figure 4.3 : Server to server data access

Another variation on this can be distributed data update where an update to data residing on server 1 triggers updates to data residing on one or more remote servers. In this more complex example, the distributed data access function must provide the capability, usually referred to as "two-phase commit," of ensuring that all updates are successfully completed before either

committing those updates or canceling them. The functionality and capability of this component will be dependent on the database vendor and its actual usefulness and importance determined by the business's requirements.

Proprietary extensions to support remote data distribution have traditionally been offered by various database vendors and serve to support, more or less, many of these data access needs. Oracle and INGRES, in particular, have long incorporated some form of distributed data access capability within their products. The problem to date, though, has been to provide such access across heterogeneous database platforms.

IBM, as the only vendor providing multiple different and incompatible SQL database products today, has published a Distributed Relational Data Architecture (DRDA) specification that offers one potential solution to this problem. Currently implemented in IBM's various platform specific Distributed Database Connection Service (DRDS) products to provide transparent distributed access to DB/2, SQL/400, SQL/DS and OS/2 Data Manager, IBM has offered this specification to the SQL access group as a basis for an industry-wide distributed processing standard.

A third approach to this problem is represented by Sybase's Open Data Services product. Open Data Services is essentially an API supported by a function library that facilitates the building of database specific gateways between SQLServer and other database products in a manner that is

transparent to the application itself. The ODS layer resides on the server and monitors all client requests, intercepting and rerouting those meant for other database servers, then passing the remote database's response back to the requester as if it had been serviced locally. Although this is a very flexible and powerful solution to a number of specific "problems," the "roll your own" distribution mechanism isn't appropriate for most application developers and its use is limited to third-party middleware vendors developing such database specific gateways as MicroDecisionWare's DB/2 gateway.

Chapter 5 Security in Client/Server

To build Client/Server system architecture, it is always important for us to consider about the network security features of that system. This is to ensure that; the system may have a secured communication between the client and the servers without any malicious attacks from the intruder and always provide the confidentiality, availability and integrity features to the data. Given these considerations, some properties of a secure communication were identified as below:

- **Secrecy:** Only the sender and intended receiver should be able to understand the contents of the transmitted message. Because eavesdroppers may intercept the message, this necessarily requires that the message be somehow encrypted so that any interceptor cannot decrypt an intercepted message. This aspect of secrecy is probably the most commonly perceived meaning of the term "secure communication", but rather restricted definition of secrecy as well.
- **Authentication:** Both the sender and receiver need to confirm the identity of other party involved in the communication-to confirm that the other party is indeed who or what they claim to be. Face-to-face human communication solves this, problem easily by visual recognition. When communicating entities exchange messages over a medium where they cannot "see" the other party, authentication is not so simple. Why, for

instance, should you believe that a received e-mail containing a text string saying that the e-mail came from a friend of yours indeed came from that friend? If someone calls you on the phone claiming to be your bank and asking for your account number, secret Personal Identification Number (PIN), and account balances for verification purposes, would you give that information out over the phone? Hopefully not.

- **Message integrity:** Even if the sender and receiver are able to authenticate each other, they also want to ensure that the content of their communication is not altered, either maliciously or by accident, in transmission.

Having established the meaning of secure communication let me introduce the three most important methods to assure a secure communication:

- User Authentication and Access Control
- Data Encryption
- Authentication

5.1 User Authentication

Users must be identified and authenticated before they are enabled access to your database. Users will be identified by Oracle, but they can be authenticated in three different ways: database, external, or enterprise authentication. Additionally, database and Web servers can be authenticated.

5.1.1 Database Authentication

Database authentication is used when a user is created and a password is specified [7]. This is a good approach for small user communities when there are no additional security products available. The other types of authentication require the reserved word `external` to be used in place of the user password.

When a user is created, a password must be selected for the user. Applying rules on a user password is called password management, and a company should have guidelines for passwords. A strong password is one that is not easily guessed, is longer than five bytes, and is not a word found in the dictionary. If the password is in a dictionary, a computer vandal might be able to guess the password by using a "brute force attack." (A *brute force attack* is one in which a computer vandal uses a `userID` and writes a program to try different passwords that are generated from a dictionary.) A password also should expire after a certain length of time and not be reused by the same user. Oracle now has the capability of providing password management when using database level authentication. This is accomplished by setting parameters on a profile and assigning that profile to a user. A profile is a database entity that specifies resource limits, and when a profile is assigned to a user, it enforces those limits on the user. A profile can be created using the Enterprise Manager or SQL*Plus. The database must have resource limits

turned on for the profile resource limits to take affect. You do this by setting the `RESOURCE_LIMIT` parameter in the `init.ora` file to `TRUE`. A profile can limit the number of sessions, the CPU usage per session, the number of CPU calls, the logical reads, the logical reads per call, idle time, and connect time. The profile can prevent computer vandals from utilizing all the resources from a computer in a denial-of-service attack.

The profile can now enforce password management rules, which are options that you can elect to be used:

- Locking of a user account-When a user has multiple failed logins, the account can be locked for a specified period of time.
- Password Lifetime and Expiration-A given password will now have a specified time limit for use and then will expire and have to be changed. A grace period will be given to a user after the password expires; if the user does not change the password, the account is locked. The database/security administrator can also set the password to an expired state.
- Password History-The password history option checks each newly specified password to ensure that a password is not reused for the specified amount of time or for the specified number of password changes. The database administrator can configure the rules for password reuse with `CREATE PROFILE` statements.

- Password Complexity Verification-Complexity verification checks the strength of a password to make it harder for a computer vandal to defeat it. The default password complexity verification routine requires that each password does the following:
 - Be a minimum of four characters in length
 - Does not equal the userID
 - Includes at least one alphabetic character, one numeric character, and one punctuation mark
 - Does not match any word on an internal list of simple words, such as welcome, account, database, user, and so on
 - Differs from the previous password by at least three characters
- Database Administrator Authentication-Database administrators require a more secure authentication scheme due to the privileged nature of their tasks (such as shutting down or starting up a database). Additional authentication can be implemented using the operating system and/or a password file.
- Operating system-If the operating system provides a way of segmenting users into groups such as UNIX or NT, Oracle will recommend DBAs be placed in a special group. This enables Oracle to have additional authentication via the group ID to know that a user is a DBA.

- Using a Password File to authenticate DBAs-A password file for DBAs is optional and can be set up using the ORAPWD password utility. The password file will restrict administration privileges to only the users who know the password and have been granted a special role. The roles are SYSOPER and SYSDBA:
 - SYSOPER enables you to perform STARTUP, SHUTDOWN, ALTER DATABASE OPEN/MOUNT, ALTER DATABASE BACKUP, ARCHIVE LOG, and RECOVER and includes the RESTRICTED SESSION privilege.
 - SYSDBA contains all system privileges with ADMIN OPTION, and the SYSOPER system privileges; it enables you to perform CREATE DATABASE and time-based recovery.

5.1.2 External Authentication

External authentication relies on an operating system or network authentication service. This places control outside of Oracle for password management and user authentication, although Oracle still identifies the user. A database password is not required for this type of login. To use this option, set the parameter `OS_AUTHENT_PREFIX` in the database `init.ora` file. This tells Oracle that any user that has the same prefix as this value is to be authenticated externally. For example, if the value is set to `ops$` and you have

two users named ops\$jones and smith, Oracle does not require a password from ops\$jones, but it does require one from smith. This parameter can be set to any prefix you want and even can be set to a null string by specifying an empty set of double quotes. The init.ora parameter REMOTE_OS_AUTHENT must be set to true (the default is false) to enable Oracle to use the username from a nonsecure connection. This keeps a potential computer vandal from masquerading as a valid user.

Network authentication is accomplished with the Oracle Advanced Security (OAS) option and can authenticate users with the following technologies:

- Network Authentication Services (such as Kerberos and SESAME)- Enable a central source for password management and can enforce single sign-on using third-party software. A user is created on each database that she will use and database privileges are assigned to that user, but the password is the reserved word external. This tells Oracle to identify the user only and enable an external source to authenticate the password. OAS uses an authentication server that has usernames, passwords, and hostnames to verify the password. If the password is authenticated, the user is enabled access to the Oracle database.
- Token Devices-A token is a physical device that a user must have to establish a connection to the database. The token could be a one-time numeric password that is generated on a device the size of a thick credit

card. This numeric password must be used in conjunction with a short numeric personal identification number (PIN). The Oracle server has an additional security service added to the configuration that is keeping track of the token's password. Another method is the challenge/response. A number is sent to the user (challenge) and the user enters the number on a device, which gives another number (response) that is used as the password.

- **Biometric Devices**-Uses a physical characteristic that is unique to the individual, currently a fingerprint scan device that can be used with OAS. The user fingerprint must first be recorded on the system, and then the user specifies the Oracle service and places her finger on the fingerprint reader. The finger placed on the reader will be compared with the fingerprint on the database.

5.1.3 ENTERPRISE AUTHENTICATION

Enterprise Authentication enables a central source for password management and can enforce single sign-on using Oracle Security Service (OSS) [8]. The user is called a global user and must be created on each database that he will use with the password globally. This tells Oracle to identify the user only and enable an OSS to authenticate the password and convey user enterprise authorizations. If the password is authenticated, the user- is enabled access to

the Oracle database. OSS interfaces with Oracle Enterprise Manager to centralize security role management and enterprise authorizations. This enables the user to have global identities that are centrally managed.

5.2 *Data Encryption*

To protect data from any malicious attack from an intruder, encryption techniques are used to scramble the data into a cipher text. After the data are encrypted then the data will be sent through network to the recipient. The recipient will then decrypt the cipher text to its original text. Currently we have two cryptography techniques, which are the **Symmetric Key Cryptography** and the **Public Key Cryptography**. In so-called symmetric key cryptography, both the sender and receiver's keys are identical and are secret. In public key cryptography, a pair of keys is used. Both Bob and Alice know one of the keys. The other key is known only by either Bob or Alice (but not both). In this section both of these cryptography methods will be discussed in detail.

5.2.1 **Symmetric Key Cryptography**

In symmetric key cryptography, both the sender and receiver are using the same for encryption and also decryption [9]. This key is a fixed-length bit string. By using this key, the sender encrypts the message lets say "THESIS",

into a cipher text form where nobody will understand it without any process of decryption is being done. This scenario is illustrated in Figure 5.1 below.

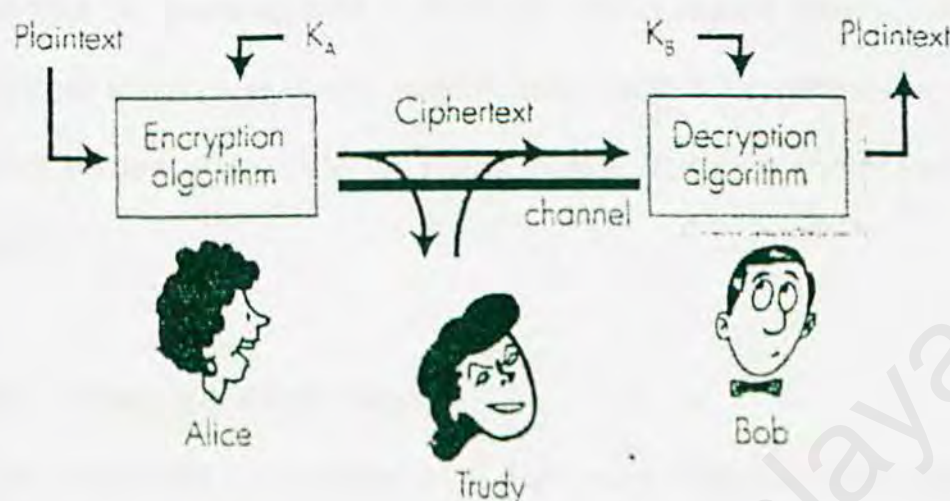


Figure 5.1 : Symmetric Key Cryptography

In Figure 5.1, Alice is the sender where else Bob is receiver. In between of them is the intruder named Trudy. So to avoid Trudy to read any of the message sent to Bob, both Alice and Bob agreed to use the same key in this transaction the message. So the sender's key, K_A and the receiver's key, K_B will be the same key.

However, the symmetric key cryptography has also several drawbacks that have to be considered. Since the "secret key" is used for both encryption and decryption, anyone who steals the key can then steal all the data that is currently or had already been encrypted, jeopardizing all present and past communications using the shared key. Because of this danger, the keys must be delivered in a protected manner such as a direct face-to-face negotiation or

a telephone call exchange. Since the privacy of all data communications is based on the integrity of the secret key, it is important to periodically replace the keys. By practicing what is known as "perfect forward secrecy," where keys are refreshed on a very frequent basis, hackers are presented a very small window of access to the system thereby ensuring a greater level of privacy.

5.2.2 Public Key Cryptography

The use of public key cryptography is quite simple. Suppose Alice wants to communicate with Bob. As shown in Figure 5.2, rather than Bob and Alice sharing a single secret key (as in the case of symmetric key systems), Bob (the recipient of Alice's messages) instead has two keys—a public key that is available to everyone in the world (including Trudy the intruder) and a private key that is known only to Bob. In order to communicate with Bob, Alice first fetches Bob's public key. Alice then encrypts her message to Bob using Bob's public key and a known (for example, standardized) encryption algorithm. Bob receives Alice's encrypted message and uses his private key and a known (for example, standardized) decryption algorithm to decrypt Alice's message. In this manner, Alice can send a secret message to Bob without either of them having to have to distribute any secret keys!

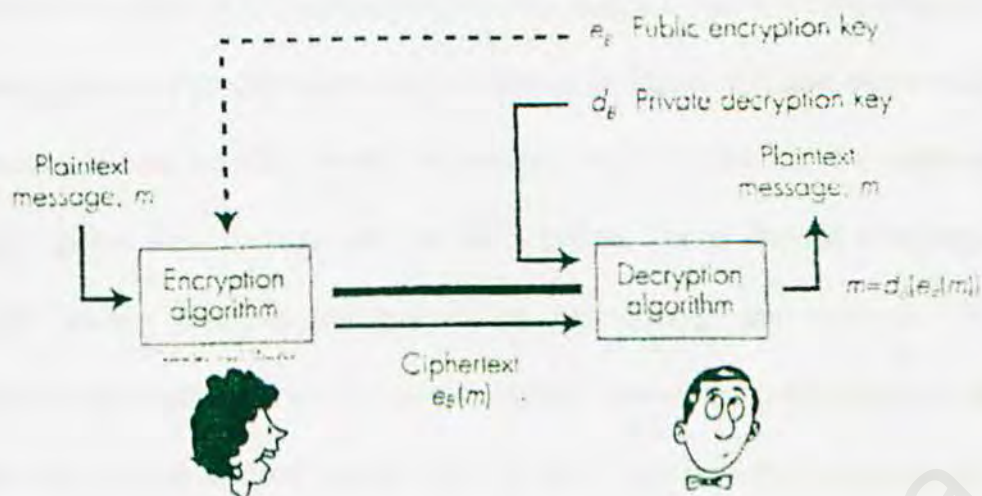


Figure 5.2 : Public Key Cryptography

Using the notation of Figure 5.2, for any message m , $d_H(e_H(m)) = m$, that is, applying Bob's public key, e_B , then the Bob's private key, d_B will decrypt the cipher text back to the original message, m . One of the most commonly used public key cryptography today is the RSA algorithm. This algorithm will be discussed in the following section.

5.2.2.1 RSA algorithm

The Rivest, Shamir and Adelman (RSA) design for a public-key cipher [10] is based on the use of the product of two very large prime numbers (greater than 10^{100}), relying on the fact that determination of the prime factors of such large numbers is so computationally difficult as to be effectively impossible to compute.

The RSA public key cryptosystem can be used for two different purposes: encryption and digital signatures. As shown in Figure 5.3, any information encrypted with the RSA private key can only be decrypted with the matching RSA public key. If Alice uses her RSA private key to encrypt a message, then anyone that has her public key can decrypt the message. This mechanism establishes a one-way trust model because the combination of the sender's private key and public key is used to perform the encryption and decryption of the message. Used in this manner, the RSA scheme (commonly referred to as an RSA Digital Signature) can be applied to many purposes, including the transport of keys or other encrypted material can be applied to many purposes, including the transport of keys or encrypted material.

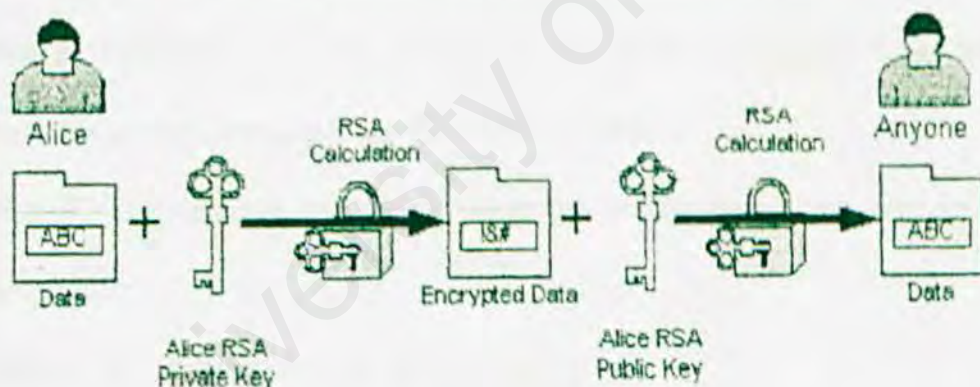


Figure 5.3 : RSA Public Key Cryptosystem

In the Figure 5.4, a request is made for a person's public key. The digital signature process starts by the sender composing a message with the person's private key. Strong digital signatures are an essential requirement for secure systems. They are needed in order to certify certain pieces of information, for

example to provide trustworthy statements binding users' identities to their public keys or binding some access rights or roles to users' identities [11].

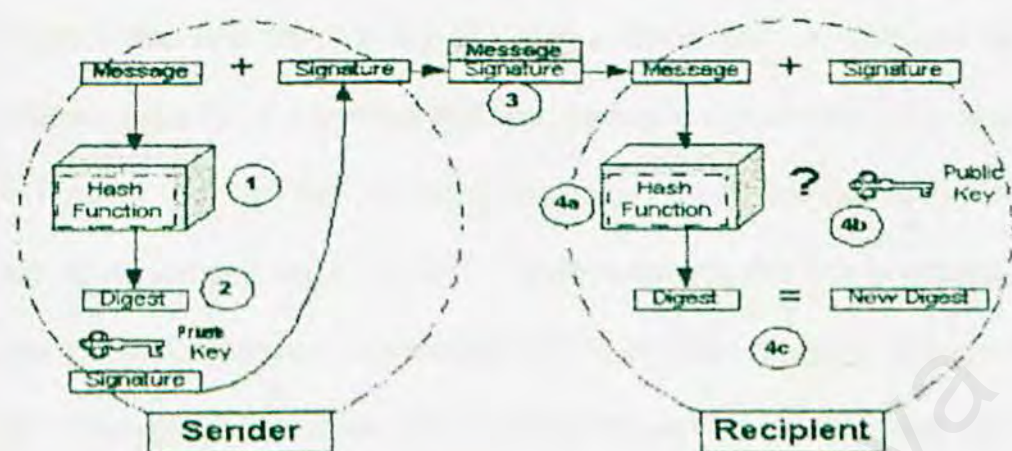


Figure 5.4 : Digital Signature Process

Step 1: A "Hash Function" (data authentication algorithm) reduces the message to a reasonably sized message called a "Message Digest."

Step 2: The sender signs the message by encrypting the message digest with their private key, creating a unique "digital signature."

Step 3: The message and unique signature are combined and sent to the recipient.

Step 4: The recipient verifies the message by:

- Generating the digest with the same hash function
- Using the sender's public key and decrypting the unique signature embedded in the message.
- Comparing the results of Steps 4a and 4b (If they match this verifies that the data was transmitted securely and unmodified)

5.2.2.2 Example Scenario for Digital Signature

Suppose that Bob wants to digitally sign a "document", m . Think of the document as a file or a message that Bob is going to sign and send. As shown in Figure 5.5, to sign this document, Bob simply uses his private decryption key, d_B , to compute $d_B(m)$. At first, it might seem odd that Bob is running a decryption algorithm over a document that hasn't been encrypted. But recall that "decryption" is nothing more than a mathematical operation and recall that Bob's goal is not to scramble or obscure the contents of the document, but rather to sign the document in a manner that is verifiable, nonforgeable, and nonrepudiable. Bob has the document, m , and his digital signature of the document, $d_B(m)$.

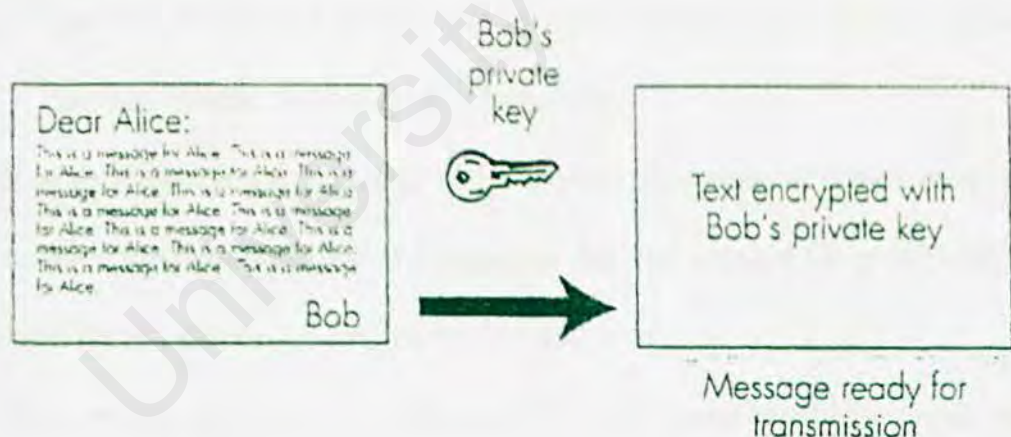


Figure 5.5 : Creating a digital signature for a document

Does the digital signature, $d_B(m)$, meet our requirements of being verifiable, nonforgeable, and nonrepudiable? Suppose Alice has m and $d_B(m)$. She

wants to prove in court (being litigious) that Bob had indeed signed the document and was the only person who could have possibly signed the document. Alice takes Bob's public key, e_B and applies it to the digital signature, $d_B(m)$ associated with the document, m . That is, she computes $[e_B(d_B(m))]$, and voila, with the flurry, she produces m , which exactly matches the original document. Alice argues that only Bob could have signed the document because:

- Whoever signed the message must have the private encryption key, d_B in order to compute the signature $d_B(m)$, such that $e_B(d_B(m)) = m$.
- The only person who could have known the private key, d_B is Bob. Therefore the only person who could know d_B must be the one who generated the pairs of keys (e_B, d_B) , in the first place, Bob. (Note that this assumes though that Bob has not given the private key to anyone, nor has anyone "stolen" the private key from Bob).

It is also important to note that if the original document, m is ever modified to some alternate form, m' the signature that Bob created for m will not be valid for m' , since $e_B(d_B(m))$ does not equal to m' .

Thus we see that public key cryptography techniques provide a simple and elegant way to digitally sign documents that is verified, nonforgeable and nonrepudiable and that protects against later modification of the document.

5.3 *Authentication*

Authentication is the process of proving one's identity to someone else. As humans, we authenticate each other in many ways: we recognize each other's faces when we meet, we recognize each other's voices on the telephone, we are authenticated by the customs official who checks us against the picture on our passport [12]. When performing authentication over the network, the communicating parties cannot rely on biometric information, such as a visual appearance or a voiceprint.

Here, authentication must be done solely on the basis of messages and data exchanged as part of an authentication protocol. Typically, an authentication protocol would run before the two communicating parties run some other protocol (for example, a reliable data-transfer protocol, a routing table exchange protocol, or an e-mail protocol). The authentication protocol first establishes the identities of the parties to each other's satisfaction; only after authentication do the parties get down to the work at hand.

5.3.1 **Authentication Protocol**

A protocol that uses public key cryptography in a manner analogous to the use of symmetric key cryptography is called the Authentication Protocol (ap5.0) (see Figure 5.6). Below is an example scenario of this protocol:

1. Alice sends the message "I am Alice" to Bob.
2. Bob chooses a nonce (a number), R and sends it to Alice. Once again, the nonce will be used to ensure that Alice is "live".
3. Alice uses her private encryption algorithm with her private key, d_A , to the nonce and sends the resulting value $d_A(R)$ to Bob. Since only Alice knows her private key, no one except Alice can generate $d_A(R)$.
4. Bob applies Alice's public encryption algorithm, e_A to the received message, that is, Bob computes $e_A(d_A(R))$. Thus Bob computes R and authenticates Alice.

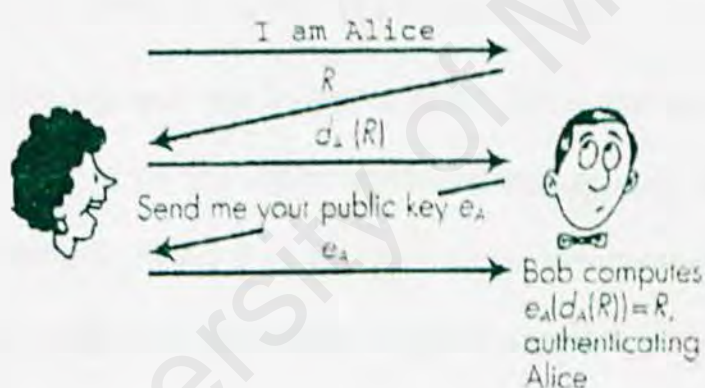


Figure 5.6 : Protocol ap 5.0 scenario

Chapter 6 Analysis on Existing System Model

Before this project started to go into the implementation phase, it will be better for me to do some analysis on the existing systems. This, not only can give me some ideas in implementing my system architecture but also giving me the chance to integrate the advantages in these existing systems into my system. Analysis on existing system might also help me to learn the weaknesses occur in these existing systems and this can avoid me from not to make the same mistakes in my system when the implementation phase comes..

6.1 *Centralized Multi-User Architecture*

An architecture approach that designs an application so that all functional and data components of the application reside and execute upon one centralized computing platform (Figure 6.1) used by multiple simultaneous users of that application. Examples of applications using this approach include order entry, accounting, manufacturing control, automated teller machine and reservation systems.

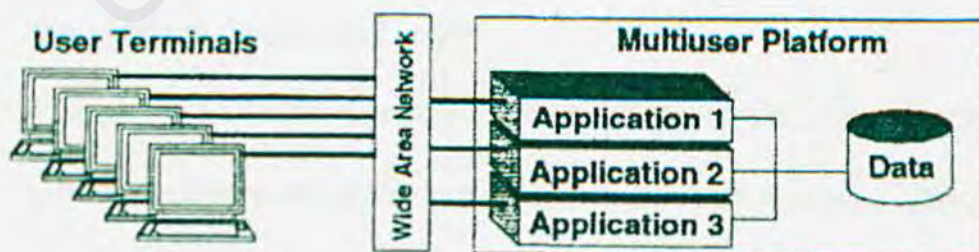


Figure 6.1: Centralized multi-user architecture

This approach is the most common architecture used for business applications based on mainframe and minicomputer technologies and its use is driven by a number of factors and the most important of which are:

- The need to leverage the costly investment in hardware, software and technical support staff represented by these centralized computers across as many applications and users as possible.
- The need to provide large numbers of simultaneous users (200 to 10,000) with reliable and stable access to one or more special-purpose applications.
- The need to provide centralized storage for very large databases shared by many simultaneous users.
- The need to minimize the amount of data that flows across relatively slow (9.6 Kbps to 1.5 Mbps) and expensive wide area networks.

6.1.1 The Strengths of Centralized Multi-user Architecture's

- This technology tends to be very stable, reliable and well supported by responsible **original equipment manufacturers (OEMs)**.
- It is capable of providing cost-effective application functionality and shared data access to thousands of users.
- A single OEM vendor can often provide all the system –level hardware, software and networking components. This “one stop shopping” capability

significantly simplifies the administration and management of the environment.

- There is a large pool of highly skilled technical staff who are available to provide technical, operational and developmental support.
- Business application software is commercially available from the OEMs and third-party vendors across a wide range of categories.

6.1.2 The Weaknesses of Centralized Multi-user Architecture

- Technologies are proprietary and with a few exceptions, generally incompatible across OEM vendors. In some cases this incompatibility even extends to different model lines from the same vendors.
- Technology within this category is expensive to acquire. Implementation costs can also be substantial as these platforms often require controlled environments with raised flooring, massive air and liquid-cooling plants, sophisticated power distribution and special-purpose fire and water damage control systems.
- These technologies require large support staffs of personnel who are highly skilled in relatively narrow technical disciplines.
- Third-party business and system applications are commercially available from only relatively limited number of vendors. License fees are generally based on hardware capacity and are expensive. For example, the mainframe

version of a functionally equivalent personal computer database management product can cost between \$100,000 to \$400,000.

- The performance characteristics of multi-user systems often result in significant upgrade costs to support small incremental increases in demand as total system capacity is approached.

After discussing a lot about the Centralized Multi-User Architecture, now we move to another system architecture called the Distributed Single-User Architecture.

6.2 *Distributed Single-User Architecture*

This approach designs an application so that all functional and data components of the application reside on a single computing platform (Figure 6.2) dedicated to the use of only one person at a time. The most common examples of application using this approach are those which we generally categorize as personal productivity aids and include word processing, spreadsheets, graphics and personal database applications.

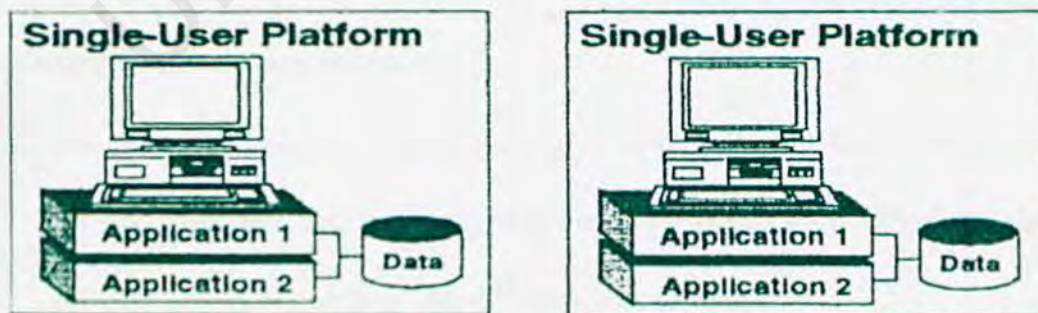


Figure 6.2 : Isolated single-user architecture

A variation of this approach, made possible by the development of local area networking technology, allows single-user applications to provide a limited form of simultaneous shared access to data across the platform interconnected by the network. Implementations of this variation (Figure. 6.3) can take many forms, depending on the capabilities of the local area network technology being used, but the application's architecture is still essentially single-user. The application "views" the data as if it was co-resident on the same platform as the application and is still designed to be independently executable on a single platform by a single user.

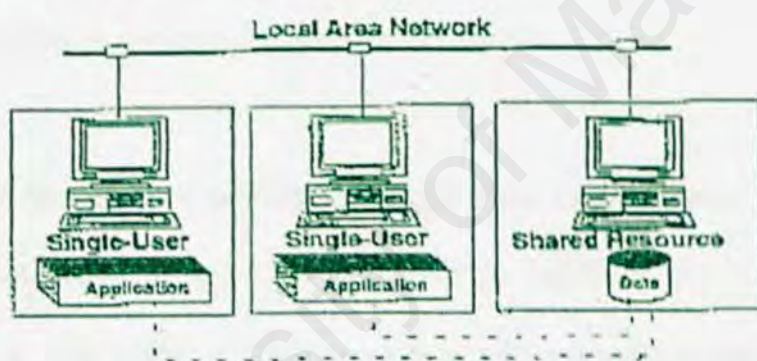


Figure 6.3 : File Server Architecture

In either its stand-alone or networked versions this approach is the most commonly used architecture for personal productivity applications. The primary factors driving its use are:

- The purpose of the application itself as a "personal" tool, used to perform a limited function for a single user without needing to simultaneously provide access to common shared repository of data.

- The ready availability of powerful, relatively inexpensive and standardized hardware platforms and operating systems.
- The commercial availability of powerful and inexpensive applications.
- The availability of fast (≥ 10 Mbps) local area network technology, making it possible to inexpensively interconnect multiple single-user computers.
- The need to cost-effectively provide small numbers of users (2 to 50) with shared access to a common repository of data that is typically much smaller than those supported by centralized multi-user applications.
- The inability of corporate MIS departments to develop and deploy smaller applications in responsive cost-effective manner.

6.2.1 The Strengths of Distributed Single-User Architecture

- The technology is based on industry-wide standards and is compatible across a wide range of OEM vendors at very reasonable cost and is also stable and reliable hardware and software.
- The alternatives available provide the ability to more exactly match processing capacity with demand and are highly scalable, allowing processing capacity to be incrementally increased in a cost-effective manner to keep pace with increases in demand.

- The hardware and system software technologies are simple to understand, use, maintain and do not require large staffs of highly skilled technical personnel to provide operational support.
- Literally thousands of reasonably priced third-party application, across hundreds of categories, are commercially available from a large number of sources.
- Third-party applications are generally designed and developed to be easily used by non-technical users.
- The user is in complete and total control of the environment.

6.2.2 The Weakness of Distributed Single-User Architecture

- The technologies and applications are targeted to support a single user. Sharing of data, applications or other resources across many users is difficult and often unreliable.
- Networking and operating system technologies are relatively unsophisticated and do not provide the control and management facilities or stability and reliability of more mature multi-user technologies.
- The environment is inherently multivendor, with one or more OEMs providing the hardware, another providing the operating system, one or more others providing networking technology and many others providing applications. This significantly increases the environment's operational and

administrative complexity and if implemented in an uncontrolled fashion, can lead to serious enterprise-wide support and reliability problems.

University of Malaya

Chapter 7 Methodologies

In this chapter, I will discuss the methods that have been used to design and plan our client/server system models. In building client/server application, actually there is a total of four phases that every system developer has to go through. These phases include the solution for system definition, solution for system development, solution for implementation and lastly, continuous improvement. Since that, our project is just in the stage of planning so I will not discuss about the last two phases here in this chapter. My target here in this chapter is just to discuss about the solution for system definition and system development. The later will be discussed when this project come to the implementation phase.

7.1 Methodologies Used

A methodology that will fit the archetype, incorporate proven total quality management and continuous improvement techniques and leverage the advantages of client/server technologies concepts will follow the pattern illustrated in Figure 7.1. This methodology is consists of three tightly coupled but concurrent legs dealing with different aspects of the solutions. These legs are the process reengineering, systems development and architectural foundation.

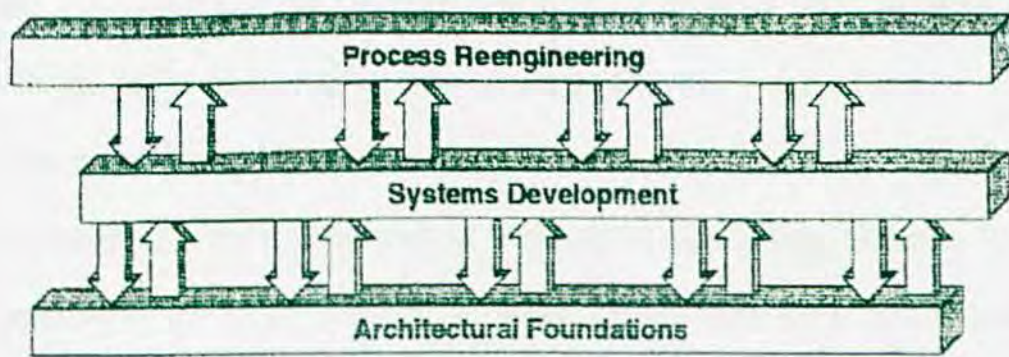


Figure 7.1 : Methodology Phases

The process reengineering leg of the project is concerned with the identifying and defining the problems to be addressed, gaining a broad understanding of the current situation and identifying the changes that must be made, defining and prototyping many different solutions until an optional solution is determined, developing the optimal solution, implementing that solution into the operational processes and then monitoring and making continuous improvements to the new processes.

The systems development leg of the project is concerned with modeling the information and workflows necessary to implement the solutions, prototyping the various solutions until an optimal solution is determined and the requirements of that solution are specified, designing and implementing a quality system to provide support as necessary and specified by the final prototype, implementing that systems solution in coordination with the new process solution implementation, and continuously enhancing and extending

the functionality of the resulting system in coordination with the ongoing continuous improvement of the new process solutions.

The architectural leg of the project is concerned with identifying and validating the tools to be used in building the application, selecting and prototyping the client, network and server technologies to be used in both prototyping and building the application, defining the manner in which the components of the application will be packaged and interact (i.e, physical system design), finalizing design, benchmarking, and implementing required data structures (i.e, physical database design), determining the distribution of data and function across the network, designing and implementing all external system interfaces, ensuring that enterprise standards are being adhered to and preparing the final production-ready deliverable for turnover to operations. During continuous improvements the architectural leg of the project will monitor and tune the database performance, optimize application resource usage where necessary and monitor and call attention to any network capacity problems.

In addition to their tightly coupled concurrency, each of these legs is internally cyclic, going through four continuous stages as illustrated in Figure 7.2.

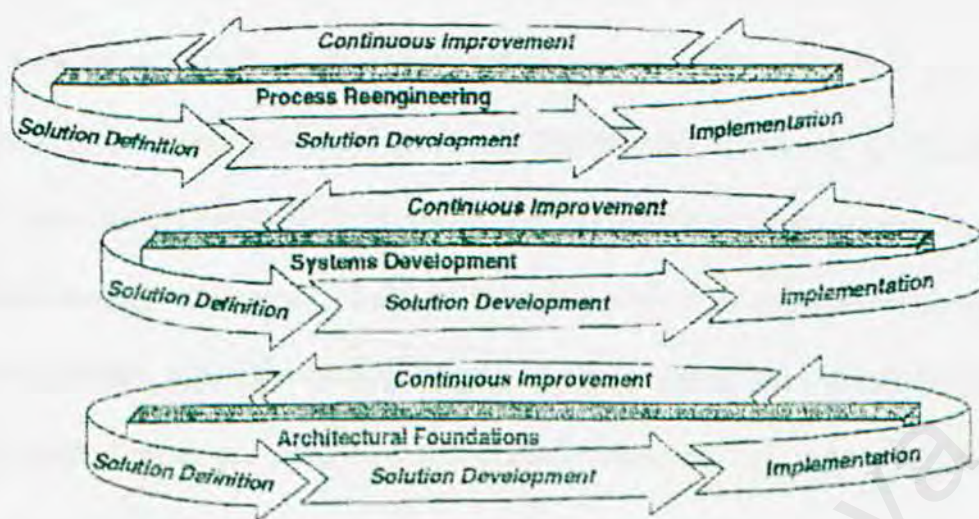


Figure 7.2 : The Natural Cycle

7.2 Solution for System Definition

7.2.1 Process reengineering

The solution definition stage of the process reengineering the project deals with problem identification and solution determination, and is broken into three primary activities, which are the activities of identifying the problems, determining root causes and defining solutions.

Identifying the Problem(s) - The first step in this stage is identification of the problem(s) that is (are) to be addressed. In given problem space there are generally a multitude of problems to be addressed. The first step in the process is to identify those root causes that are key to providing the greatest benefit with the least expenditure of effort. This step begins by gathering as

much data as possible concerning existing problem potential opportunities for improvement, and builds a complete understanding of the current environment, workflows, and processes. The primary deliverable of this step is a concise definition of the problems, and opportunities for improvement.

Determining root causes - Each problem is examined in terms of the current environment and all possible causes of the problem are identified and verified. Causes are described and classified into one of eight categories: process, equipment, personnel, material, management, data, metrics, or physical environment Relationships between the causes identified and the recognized effects of those causes are identified and quantified by strength of association. On the basis of this analysis, determine the root causes of the most significant problems. Prioritize the root causes in terms of their contribution to the significant problems. Identify those root causes that are within the boundaries of the work team's authority to address. The primary deliverable of this step is a concise identification, description, and prioritization of the relevant root causes.

Defining solution(s) - The work team evaluates a wide range of alternative solutions to address the significant root causes. The most promising of the alternatives are prototyped for their suitability and ability to address the root causes. As each alternative prototype fails, lessons are learned about what worked and what didn't and other alternatives are modified to incorporate

these lessons and prototyped again. This process continues until a solution or set of solutions is identified as those necessary to correct the root cause(s) of the problem(s). Surviving solutions are evaluated in terms of ease of implementation, negative side effects and by-products, consequences of implementation failure, ability of the work team to implement the solution, cost, and time to implement. The optimal solution is identified and a set of requirements for implementing that solution(s) documented. The primary deliverable of this step is a complete description of the solution(s) to be implemented and an action plan for their full development and implementation within the workplace.

7.2.2 System Development

The solution definition stage of the system development leg of the project is initiated with or very shortly after the beginning of process reengineering and proceeds in coordination with and support of those activities. It is broken into three basic activities.

Recovering current system design - The first step in this stage is basically a review of the current systems support being provided within the problem space and documentation of the current functions, technologies, and information structures being used by or potentially impacted by the processes under study. The primary deliverable of this step is initial population of a

design repository with descriptions of information currently maintained and used.

Determining user needs - This step deals with gaining a thorough understanding of the problem space and the root causes being identified as potential opportunities for improvement. A model of the current processes and workflows and an initial information model of the main business objects identified as within the problem space is produced to verify understanding and assist the reengineering effort in quantifying root causes and identifying potential solutions. Initial prototypes of human/system interfaces are prepared in coordination with solution prototyping to determine feasibility, usability, and suitability of system support for the solutions being analyzed.

Defining functional requirements - This step deals with completion of the information model, development of a fully specified process or workflow model of the optimal solution, a completed prototype of the automated functions determined as necessary to support the optimal solution, a fully documented set of requirements describing the functionality to be provided by automated systems, and a detailed action plan indicating how the system to meet those requirements will be developed in coordination with the solution implementation plan.

7.2.3 Architecture

The solution definition stage of the architectural leg of the project is initiated with or very shortly after the beginning of process reengineering and proceeds in coordination with and support of those activities. It is broken into three basic activities.

Reviewing current technologies - The first step in this stage is basically a review of the technologies in use. Current workstation hardware and software are cataloged and evaluated for obsolescence and compliance with enterprise standards. Network connectivity and current system platform technologies are assessed and evaluated against enterprise plans and directions, and potential external interfaces are identified and researched.

Determining appropriate technologies - This step compares currently used technologies against enterprise directions and plans, evaluates the potential of both current and potential replacement technologies to meet the requirements of the optimal solutions being developed within the reengineering leg, prototypes viable solutions, and selects and installs appropriate technologies to meet the needs of those solutions.

Development tool selection - In coordination with the system development leg, potential development tools and technologies are evaluated against the projected needs of the optimal solutions being defined within the reengineering leg and appropriate development tools and technologies are

selected and installed. Where necessary, system development personnel are trained in the effective use of the tools selected. Relevant enterprise standards are identified and reviewed.

7.3 Solution for System Development

7.3.1 Process reengineering

The solution development stage of the process reengineering leg of the project deals with developing pilots of the new workflows and processes required by the defined solutions, preparation of documentation and training programs for implementation of the solutions, and determination of the quality metrics that will be used to judge the effectiveness of the optimal solutions after their implementation. This stage consists of three primary activities.

Piloting the solutions - The first step in the piloting of the optimal solutions is to verify their adequacy and completeness. Adjustments are made as they are identified and, where necessary, are communicated to the systems development leg for inclusion into the system design.

Document evaluation metrics - Review the initial problem and root cause documentation and quantify the improvements those are to be expected through implementation of the optimal solutions. Develop and pilot the test methods to be used to gather post implementation quality statistics.

Training preparation - Identify specific areas needing educational and training support to provide transition assistance to personnel impacted by the optimal solution changes. Develop appropriate classroom curriculum, prepare training materials, develop and qualify designated mentors, and finalize training action plan.

7.3.2 System Development

The solution development stage of the system development leg is concerned with implementation of the physical database structure, construction and testing of the system in accordance with the architectural specifications provided by the architectural leg, development and testing of the conversion functions required to facilitate migration to the new system, and execution of the migration and conversion plan. It is broken into four basic activities.

Building physical database structure - The first step in this stage is implementation of the final physical database structure, coding and testing of all server-based application functions (stored procedures, triggers, user-defined functions, etc.).

Constructing the application - This activity involves the construction and unit testing of all modules of the application's architecture as designed within the architectural leg.

Testing the application - This activity involves the integration (i.e., module to module), system (i.e., function to function), and performance (i.e., benchmarking) testing of the application. Final usability, functionality, and acceptance testing is performed with the initial set of designated mentors in coordination with the reengineering leg.

Constructing and testing conversion functions - This activity, which often runs concurrently with the other development activities, task-codes and tests the conversion functions necessary to effect migration of data and functionality from the previous system (if any) to the new system.

7.3.3 Architecture

The solution development stage of the architectural leg is concerned with the physical design and specification of the application and database; the physical design, construction, and testing of all interfaces to external systems; the installation, integration, and testing of any new workstation, server, or networking technologies being used; and the development and documentation of a complete conversion plan. It is broken into six basic activities.

Completing the physical database design - Implement the final logical database design as specified. Prototype and benchmark the application's known high-volume and heavy-use data access patterns. Generalize logical database design, and make adjustments in planned redundancy, indexes, and

physical storage allocation's until the performance targets of the application are achieved. Complete physical database design and specify all database resident control and API mechanisms (stored procedures, triggers, etc.). Determine and map physical data distribution over the network and, if necessary, specify synchronization mechanisms.

Completing the application architecture - Package application functions into modules and subtasks, define intermodule message format standards and valid process flows, determine distribution of application functions across the network, and identify, define, and specify reusable resources. Verify adherence to appropriate enterprise standards and procedures.

Developing the conversion plan - Identify and define existing data sources that will migrate to the new application. Map existing data sources to their equivalents in the new database and specify all conversion, synchronization, and timing considerations. Determine the nature of the conversion (i.e., parallel vs. immediate cutover) and document a fall-back (backup) plan in the event of catastrophic conversion failure.

Developing external interfaces - Design, construct, and test all functions required interfacing with external data and application resources as required in support of the optimal solutions.

Preparing for production turnover - Prepare production or operational turnover documentation and operational procedures in accordance with existing enterprise standards for operational systems.

Installing and integrating technology - Coordinate the installation, integration, and testing of any new workstation, software, server, or network technologies required implementing the optimal solutions.

8.1. System and Application Design

Software design is a process of defining and describing the overall architecture for a software system. It involves identifying the major components of the system, their interactions, and the data flow between them. Software design is the first step in the process of transforming requirements into a concrete representation of the system's structure and behavior. It is also includes lower level work such as detailed specifications of data structures and algorithms within the identified components.

In this project, the purposes of this phase are:

- To transform requirements into working system.
- To determine a set of components and intercomponent interfaces that will be a specified set of requirements.

Chapter 8 System Design

In chapter 7, I have introduced some methods in how to develop a client/server application. So by studying these methods, I have come to a conclusion on how my client/server application will be and what are the modules I wanted to include in my client/server application. In this chapter, I will introduce in detail about my system models and the modules in this system.

8.1 System and Application Design

Software design is a process of devising and documenting the overall architecture for a software system. It includes identifying the major components of the system, specifying what they are to accomplish and establishing the interfaces among the components. Design is the first step in the process of transforming the requirements into a close representation of the eventual function software. It is also includes lower work such as detailed specification of data structures and algorithms within the identified components.

In this project, the purposes of this phase are:

- To transform requirements into working system.
- To determine a set of components and intercomponent interfaces that satisfies a specified set of requirements.

- To change the abstract logical model to concrete physical implementation.

Under the system functionality design, we will look at the system architecture and the modules in this system.

8.2 *System Architecture*

Before I proceed any further into my client/server system model. There is a need for me to introduce the two of the most common client/server architecture. These two architectures will be the 2-tier architecture and the 3-tier architecture. The 2-tier architecture splits the processing load into two [13]. The majority of the application logic runs on the client, which typically sends requests to a server-resident database. Meanwhile, a 3-tier architecture splits the processing load in between the client and the server. Which means the application is stored in a single application server rather than it resides inside the client workstation (see Figure 8.1)

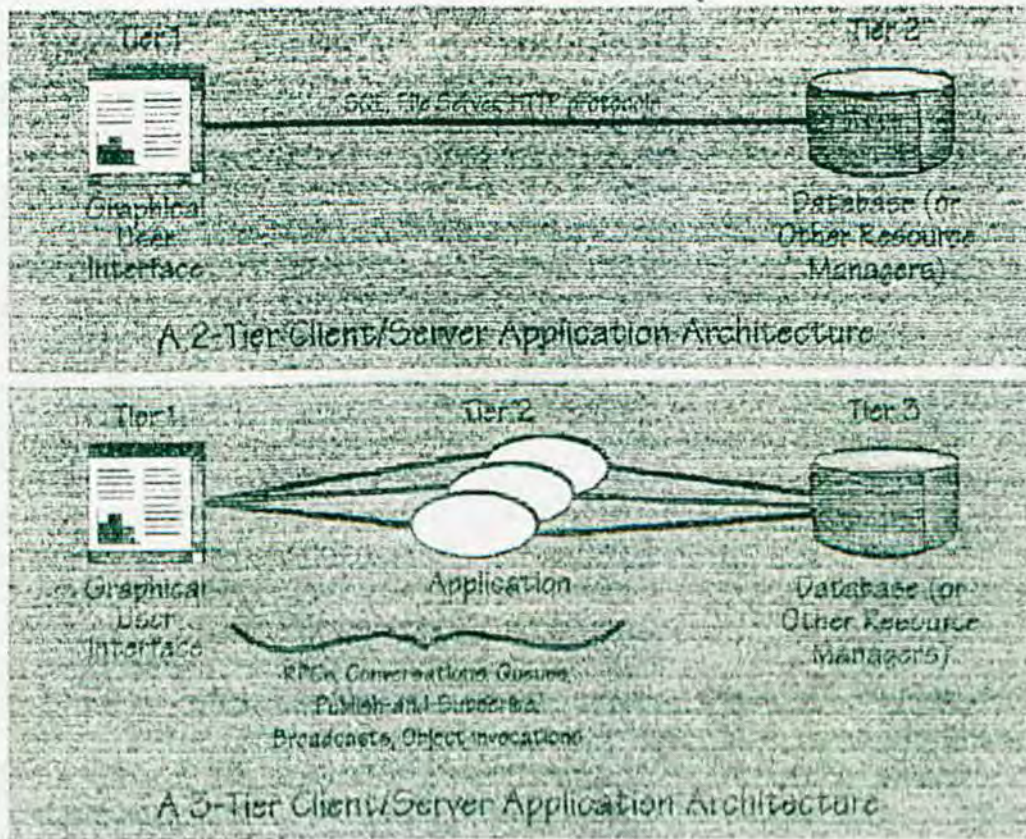


Figure 8.1 : 3-tier and 2-tier client/server architecture

3-tier architecture is the new growth area for client/server computing because it meets the requirements of large-scale Internet and client/server application. 3-tier applications are easier to manage and deploy on the network – most of the code runs on the servers, especially with zero-footprint technologies like Java applets. In addition, 3-tier applications minimize network interchanges by creating abstract levels of service. Instead of interacting with the database directly, the client calls application logic on the server. The applications logic then accesses the server on behalf of the client. 3-tier substitutes a few server calls for many SQL queries and updates, so it performs much better than 2-

tier. It also provides better security by not exposing the database schema to client and by enabling more fine-grained authorization on the server. Below are the comparisons between 3-tier and 2-tier.

Subjects	2-tiers	3-tiers
System administration	Complex (more logic on the client to manage)	Less complex (the application can be centrally managed on the server, application programs are made visible to standard system management tools)
Security	Low (data-level security)	High (fine-tuned at the service or method level)
Encapsulation of data	Low (data tables are exposed)	High (the client invokes services or methods)
Performance	Poor (many SQL statements are sent over the network; selected data must be downloaded for analysis on the client)	Good (only service requests and responses are sent between the client and server)
Scale	Poor (limited management of the client communications links)	Excellent (concentrates incoming sessions; can distribute loads across multiple servers)
Application reuse	Poor (monolithic application on client)	Excellent (can reuse services and objects)
Ease of development	High	Getting better (standard tools can be used to create the clients. And tools are emerging that you can use to develop both clients and server sides of the application)
Server-to-server infrastructure	No	Yes (via server-side middleware)

Legacy application integration	No	Yes (via gateways encapsulated by services or objects)
Internet support	Poor (Internet bandwidth limitations make it harder to download fat clients and exacerbate the ready noted limitations)	Excellent (thin client are easier to download as applets or beans; remote service invocations distribute the application load to the server)
Heterogeneous database support	No	Yes (3-tier applications can be use multiple databases within the same business transaction)
Rich communication choices	No (only synchronous, connection-oriented RPC-like calls)	Yes (support RPC-like calls, but can also support connectionless messaging, queued delivery, publish-and-subscribe, and broadcast)
Hardware architecture flexibility	Limited (you have client and a server)	Excellent (all 3-tiers may reside on the different computers, or the second and third tiers may both reside on the same computer, with component-based environments, you can distribute the second tier across multiple servers as well)
Availability	Poor (can't fail over to backup server)	Excellent (can restart the middle tier components on the other servers)

From the table of comparison above, it is now for sure that I will choose the 3-tier architecture for our client/server system and the system that I have planned to develop is roughly illustrated in Figure 8.2.

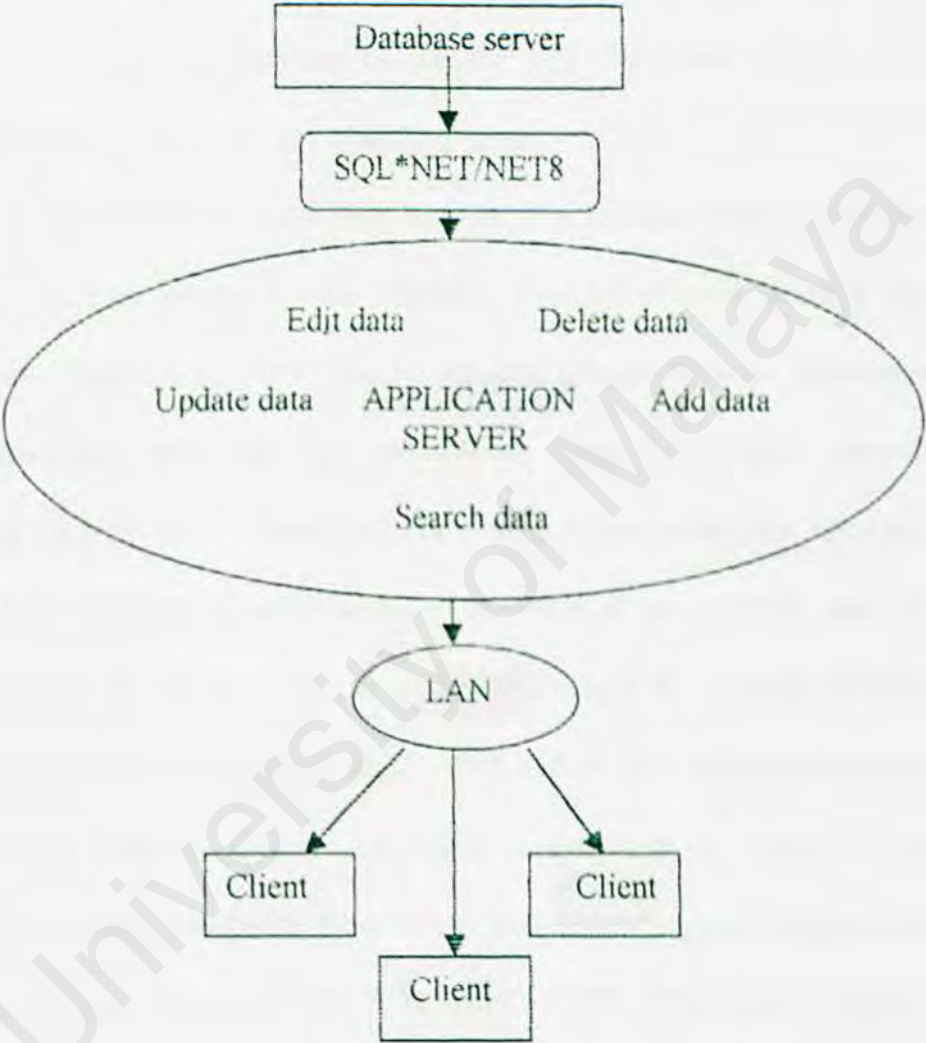


Figure 8.2 : System model

The model of this system is designed in this way because this is the design where 3-tier client/server architecture is look like. In this system models, the server will be an Oracle database server, which will be connected with the

client via a TCP/IP network transport protocol, and the data link protocol that will be used here is the Point-to-Point Protocol. Meanwhile, the middleware used here will be the SQL*NET/NET8 which is an Oracle's client/server middleware product that offers transparent connection from client tools to the database, or from one database to another. SQL*Net/ Net8 works across multiple network protocols and operating systems.

The SQL*NET/NET8 is used here because this product contain rich core network services including data transport, network naming, security and transaction monitoring. These Oracle network product form an abstraction layer, insulating users and user applications from the physical network, allowing heterogeneous, distributed computing across computers regardless of vendor, operating system, hardware architecture, or network topology. Application will work as written on an AS-400 with LU6.2 network protocol, on a Token-Ring network, or on an HP-9000 with TCP/IP network protocol n an Ethernet network. Oracle SQL*NET is available on virtually every platform supported by Oracle, from PCs to mainframe, supports almost every network protocol including TCP/IP, Novell SPX/IPX, IBM LU6.2, NetBIOS, DECNet and AppleTalk.

Since that Visual Basic 6.0 is able to intercommunicate with the database server by using SQL language and it is said to be easy to understand, this language is used to develop the system's application.

8.3 *Module Design*

Data flow diagrams (DFD) depict the broadest possible overview of system inputs, process and outputs. It able to conceptualise how the data moves through the organization, the process or transformation that the data undergo Data Retrieving Module and what the outputs are.

8.3.1 **Data Retrieving Module**

In this module, database administrator can view and make changes to the data via remote-access from any of the client, which runs the application written by the network operator (see Figure 8.5). But for the user, they can only search for the data and view it without any permission for them to do any changes to the data in the database server (see Figure 8.6). The procedure of this module is simple.

- User login to the application via a window-based form (see Figure 8.4).
- User type in their name and password
- User submits form.
- Output based on what the user requests for.

In this query form, user can choose from which category they want to view the data. The categories that we have are student, staff, facilities and activities. The DFD in Figure 8.3 shows in more detail about how the data flows in this module.

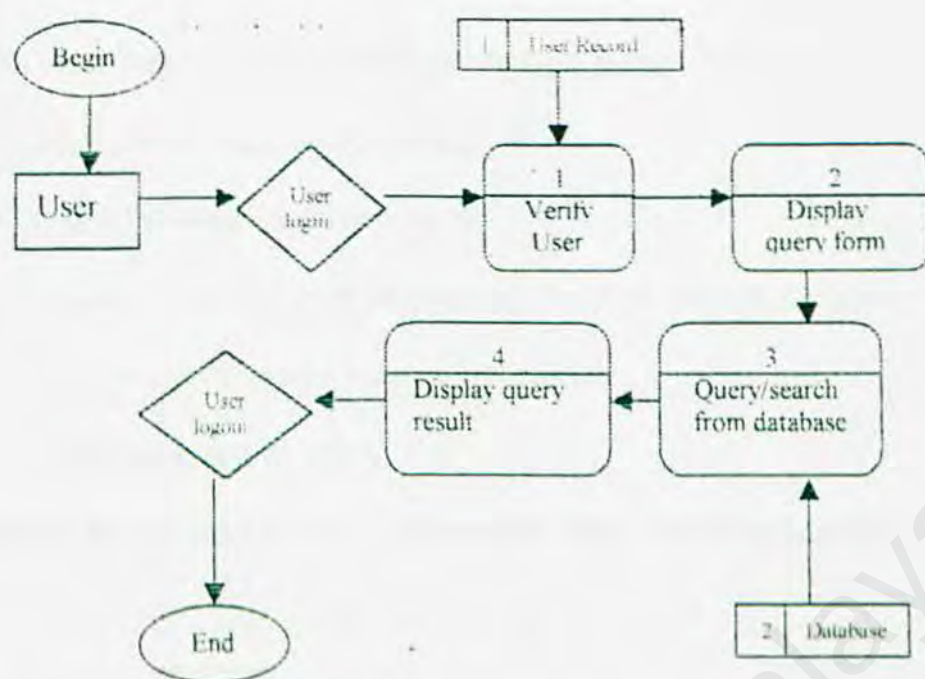


Figure 8.3 : Data Flow Diagram (DFD)

8.4 User Interface Design

User interface can be very tricky to design because different people have different styles of perceiving, understanding and working. For example, one of the users may interpret a button with a binocular picture as a “zooming button” while another may interpret it as a “searching button”.

Therefore planning a user interface is always an important and tricky job for the application developer. So, a careful planning have to be done before a real interface can be fully developed to avoid any confusion among the users.

An interface should meet the objectives of effectiveness, accuracy, ease to use, consistency, simplicity and attractiveness.

To meet these objectives, there are four guidelines that can be followed by the designer and these guidelines are:

- Make the input screen easy to fill
- Ensure that screen meet the purpose for which they are designed
- Design form to assure accurate completion
- Keep the screen attractive

Below are the input screens of the module, Data Retrieving Module:

Username:

Password:

Figure 8.4 : Login form

Select a category:

Figure 8.5 : Administrator query form

Select a category:

Search

Figure 8.6 : User query form

8.5 *Designing the Effective*

Output is information delivered to users through the information system by way of intranet, extranet, or the internet. Output can take many forms: the traditional hard copy of printed reports, soft copy such as computer screen, microforms and audio output.

Since useful output is essential to ensuring the use and acceptance of the information system, there are several objectives that the systems analyst must focus on when designing output. There are six objectives for designing output:

- Design output to serve the intended purpose
- Design output to fit the user
- Deliver the appropriate quantity of output
- Assure that the output is where it is needed.
- Provide the output on time.
- Choose the right output method.

In this project, screens are chosen for output because of the following reasons.

- Interactive.
- Quiet.
- Takes advantage of computer capabilities for movement within the databases and files.
- Good for frequently accessed, ephemeral message.

There are four guidelines to facilitate the design of screens:

- Keep the screen simple.
- Keep the screen presentation consistent.
- Facilitate user movement among screens.
- Create an attractive screen.

8.6 *Problems Faced*

During the duration of finishing this report, I have faced some problems such as below:

- Collecting information about the client/server architecture.
- Understanding the features in client/server architecture.
- Understanding the steps in developing and configuring computer network.

Luckily, all these problems is all settled by now and the solutions that I have used to solve these problems are:

- Searching information from the books and Internet.
- Having discussion with my supervisor, Pn. Hannyzzura Pal and my thesis partner Mr. Gan Hong Fee
- Reading articles and reference books about the computer networking and client/server architecture.
- Referring to past years thesis work.

8.7 Expectation on Project Output

After a long term of planning on this project, at last I managed to come out with this project proposal. In my expectation, if nothing turns wrong in between the implementation phase, this client/server system model will finished in time and it will be full of security features and shall work well as planned. This system models will have the capability to send, data in and out from the server smoothly and the most important are securely in a short response time. This system model will also have a user-friendly interface to make the interaction between the user and the system application becomes comfortable and easy.

Chapter 9 Configuring Client And Server

This chapter will discuss on how to make direct connection to between two computers in Microsoft Windows 2000. Through LAN connection, a computer can be configured either to act as a host (server) or a guest (client). I will start this chapter by looking into the steps in setting up a client computer and then follows by the server.

9.1 *Setting Up Client in Windows 2000*

Before a computer can act as a client to retrieve information from a server, we need to make a few connection settings and configuration steps. Here are the steps on how to set up this connection:

1. Under Start Menu on the control bar, select Setting then, Network and Dial-Up Connections and from there choose Local Area Network.
2. A form as below should appear in front of the screen. Click "Properties".

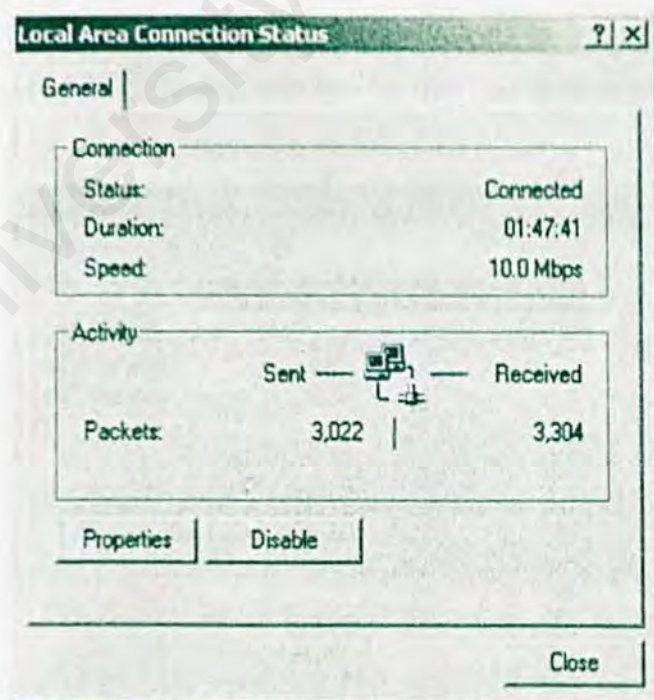


Figure 9.1: Local Area Connection Status Form

Click "Install" from the form below to add some new features for this connection.

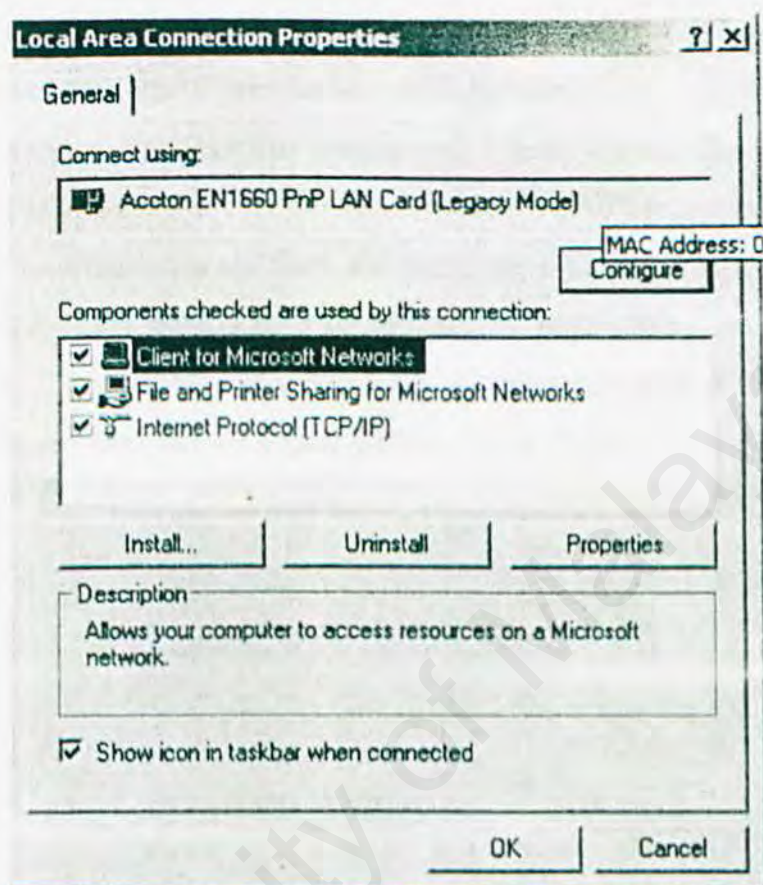


Figure 9.2: Choosing features for this new connection

3. Select the type of components to be installed

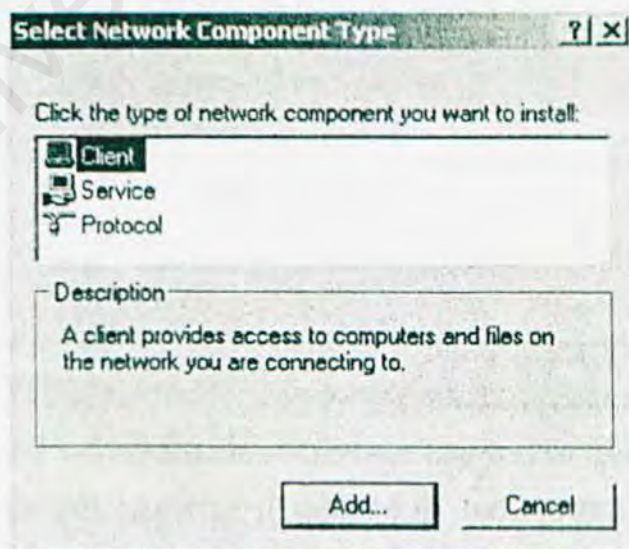


Figure 9.3: Select the network component type

4. After choosing the Client component, select the corresponding Network Client depends on what platform is your computer in. For our project, I have chosen the "Client for Microsoft Network".
5. For Service and Protocol component, I have chosen the File and Printer For Networks and Internet Protocol (TCP/IP) respectively.
6. After all these steps are done, the next step is to assign an IP address for this client computer. This is done by highlighting the Internet Protocol component and then clicked the properties button. A form as shown as below should appear in front of the screen.

Internet Protocol (TCP/IP) Properties ? X

General

You can get IP settings assigned automatically if your network supports this capability. Otherwise, you need to ask your network administrator for the appropriate IP settings.

☐ Obtain an IP address automatically

☒ Use the following IP address:

IP address:	100 . 100 . 100 . 100
Subnet mask:	255 . 255 . 255 . 0
Default gateway:	

☐ Obtain DNS server address automatically

☒ Use the following DNS server addresses:

Preferred DNS server:	
Alternate DNS server:	

Advanced...

OK Cancel

Figure 9.4: Initialize the IP address and Subnet Mask.

I have given this client an IP address of 100.100.100.100 and the Subnet mask as 255.255.255.0.

9.2 Setting Up Server in Windows 2000

The steps in setting up a server in Windows 2000 are the same with the client. Except that, the IP address is set to another value such as 100.100.100.101. The Subnet Mask is still the same with the one set in the client computer.

9.3 Installing The Oracle Client

After giving both the server and client their specific IP address. The Oracle Client software is installed into the client computer. For this project, the Oracle Client Version 8.1.6 is installed through the Oracle's Universal Installer, which is a user-friendly interface that helps the user to complete their installation easily.



Figure 9.5: Oracle's Universal Installer

Just follow the instructions given during the installation; the client software will be then successfully installed.

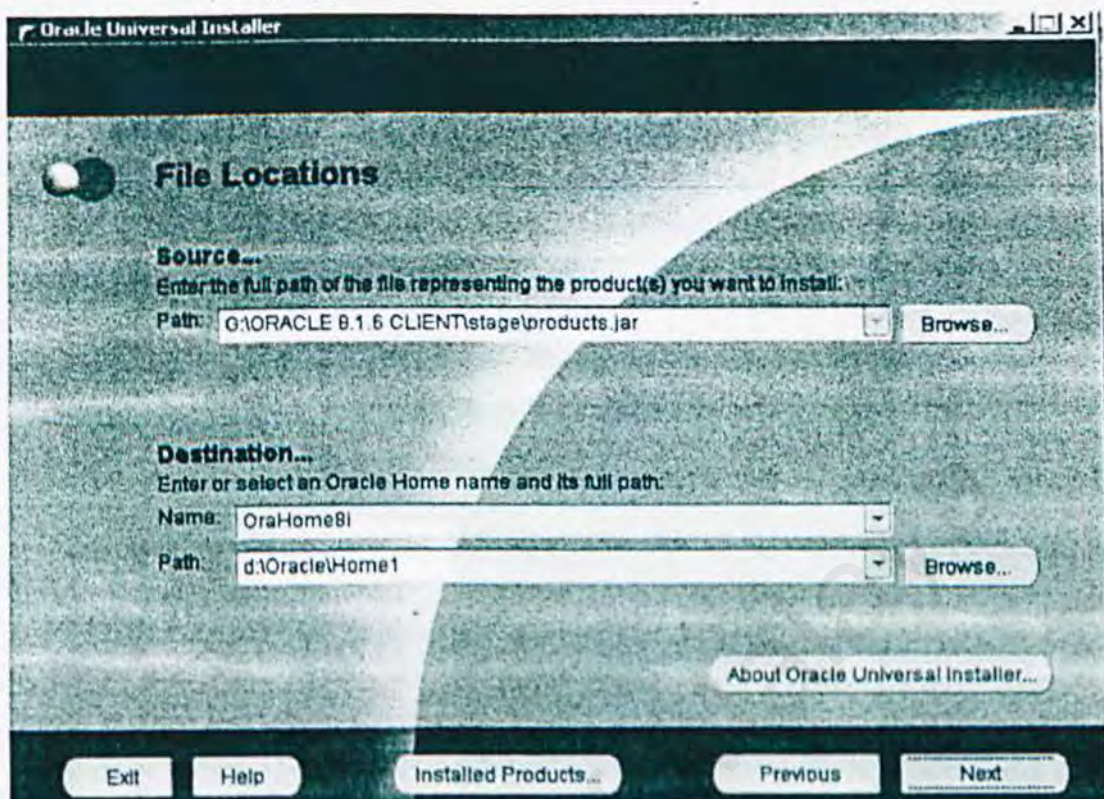


Figure 9.6: Specifying the location for this installation.

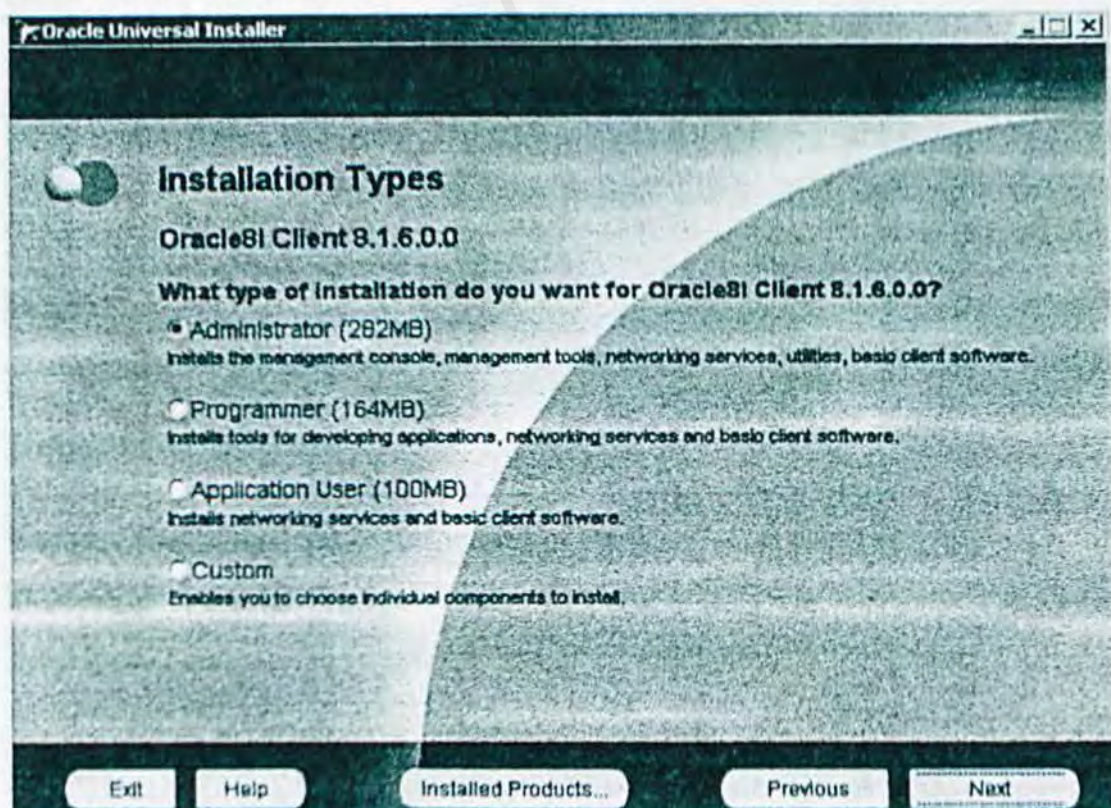


Figure 9.7: Choosing the installation type.

After the installation of the Oracle Client is finished. The next phase is to configure it, so that it is able to communicate with an Oracle Server.

9.4 *Configuring The Oracle Client*

To configure the Oracle Client, I have chosen to use the new feature added in the Oracle Client 8.1.6 that is the Net8 Assistant as shown in Figure 9.7.

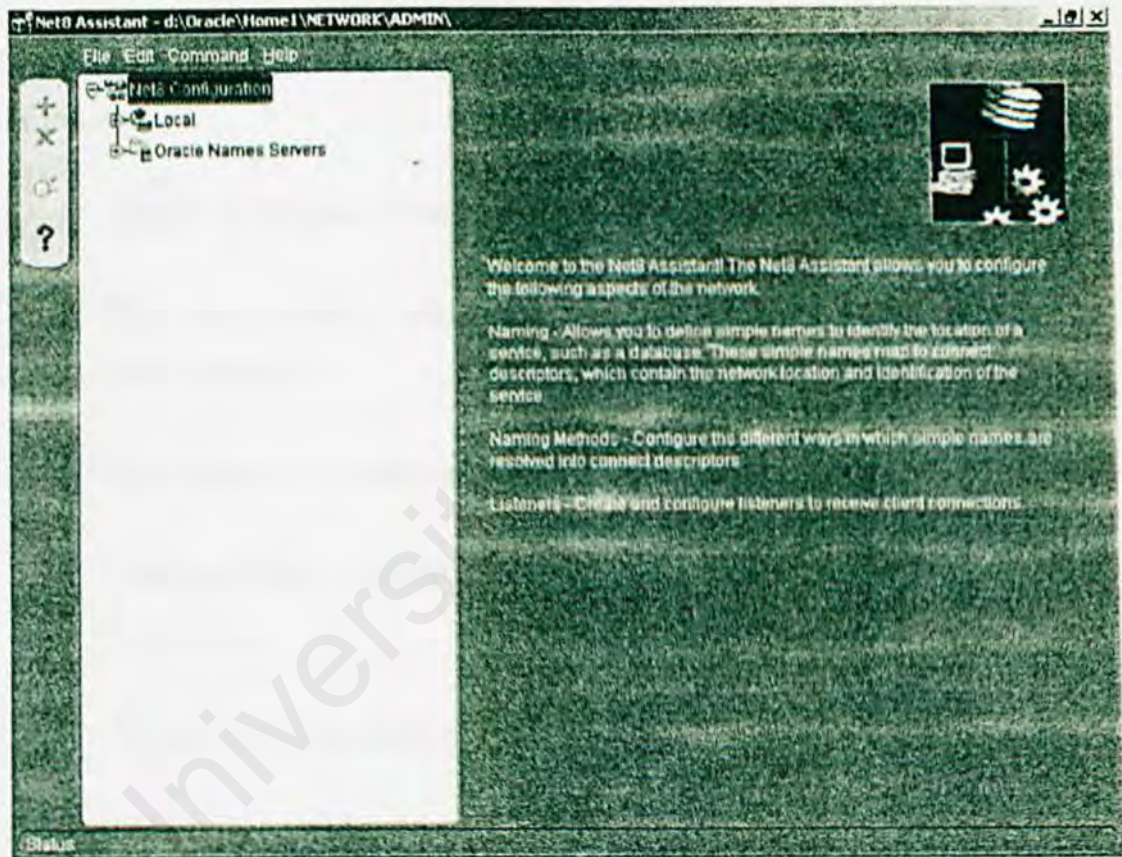


Figure 9.8: Net8 Assistant

The very first thing to do in the process of configuring the client was to choose the type of naming method for this client to resolve a net service into the information required to connect to a specific database. In this project, I have chosen the local naming method.

9.4.1 Local Naming

Local naming refers to the method of resolving a service name to a network address by using information configured on each individual client. Much like an address book, this information is entered in a local naming configuration file called TNSNAMES.ORA.

Establishing a Connection Using the Local Naming Option

The process for establishing a client session using the local naming option is as follows:

1. The client initiates a connect request providing a service name.
2. The service name is resolved to a network address configured in a local naming file.
3. Net8 makes the connect request to the address provided.
4. A network listener receives the request and directs it to the database it is servicing.
5. The server accepts the connection.

Configuring Local Naming

To configure local naming, proceed as follows:

1. Verify that "TNSNAMES" is listed in the field of selected naming methods in the client profile. If it is not, use the Oracle Net8 Assistant to edit the profile.
2. Verify that service names are correctly mapped to their appropriate network addresses in a local naming configuration file

(TNSNAMES.ORA). If they are not, you may use the Oracle Net8 Assistant to add or modify service names.



Figure 9.9: Adding the Naming Method for Oracle Client

As I have noted earlier, the naming method chosen for this connection is the Local Naming Method so I have selected the TNSNAMES from the available methods as shown in Figure 9.9. For the others field such as Oracle Names and External, I just left it without any configuration needed for those fields. This is because those fields are made just for others naming method and not for Local Naming Method.

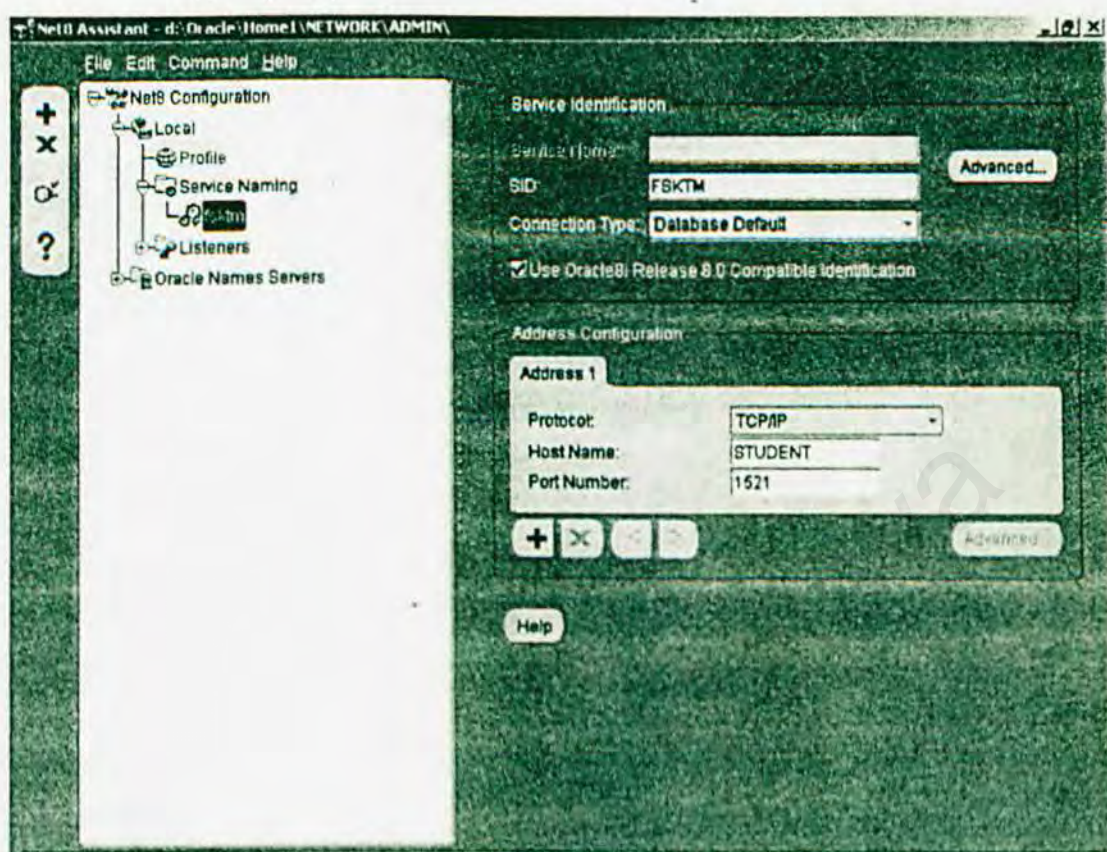


Figure 9.10: Configuring the Local Naming Method.

As shown in Figure 9.10 I have filled in the Hostname field with the Oracle Server name, STUDENT and the Port Number as 1521, which is a default value for TCP/IP protocol. Not to forget also about the security features, I have chosen to use the default security setting for this project because of it's flexibility and maximum protection for the transaction of the information from either from the client or from the server. Figure 9.10 shows the default setting of the security features.



Figure 9.11: Security features in Net8

9.5 Configuring The Oracle Server

Actually, there are lesser configuration works in the Oracle Server. What I have to do is only to configure the Listener and that's all. From the **Oracle Net8 Administrator's Guide Release 8.0** it is clearly stated, "Before Oracle8 and Oracle7 servers can receive connections from Net8 clients, you must first start a network listener on the server node. Start a listener and it will listen on a specific default address (Port 1521, TCP/IP, and interprocess communication addresses with KEY=PNPKEY). In a simple TCP/IP network not using Oracle Names, no further configuration is required." Because our project is using the simple TCP/IP network, there is no further configuration needed. The Listener configuration interface is shown in Figure 9.12.



Welcome to the Net8 Configuration Assistant.
This tool takes you through the following
common configuration steps:

Choose the configuration you would like to do:

- ☒ Listener configuration
- ☐ Naming Methods configuration
- ☐ Local Net Service Name configuration
- ☐ Directory Service Access configuration

Cancel

Help

Back

Next >

Figure 9.12: Listener Configuration

From the menu, Listener configuration is chosen and then when the next form turned up the Add option button is then clicked as shown below.

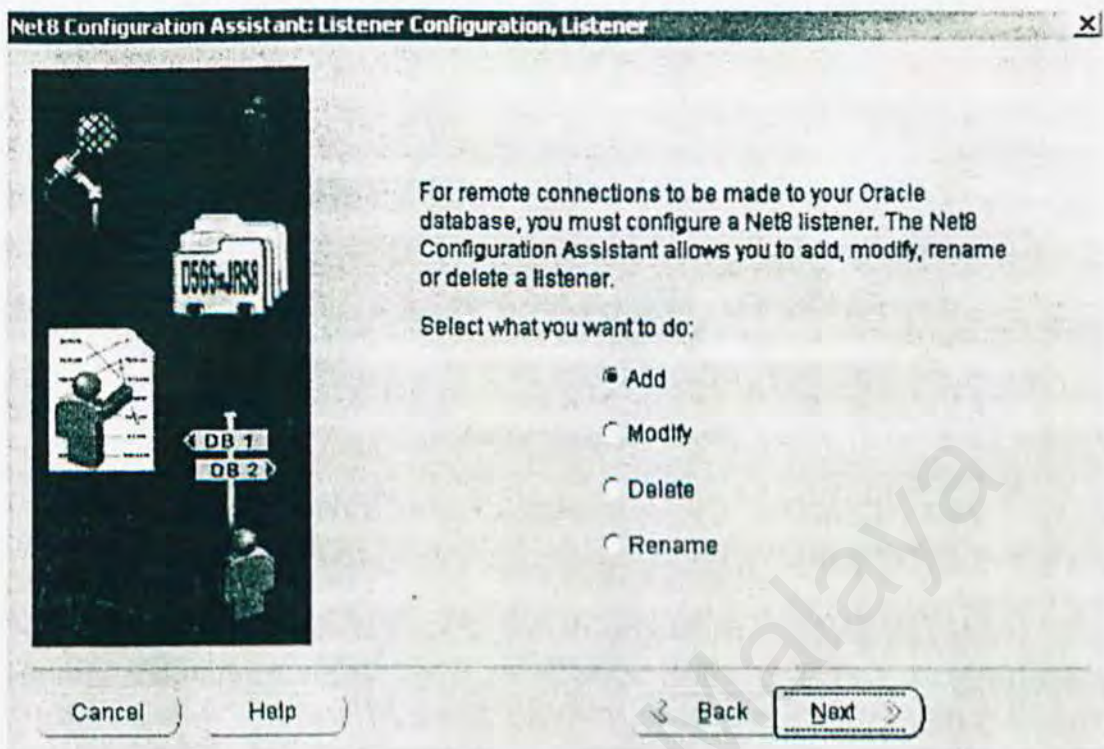


Figure 9.13: Add a new listener

After the listener is added, a name has to be given to this listener. In this example, I will give the name of this listener as "LISTENER".

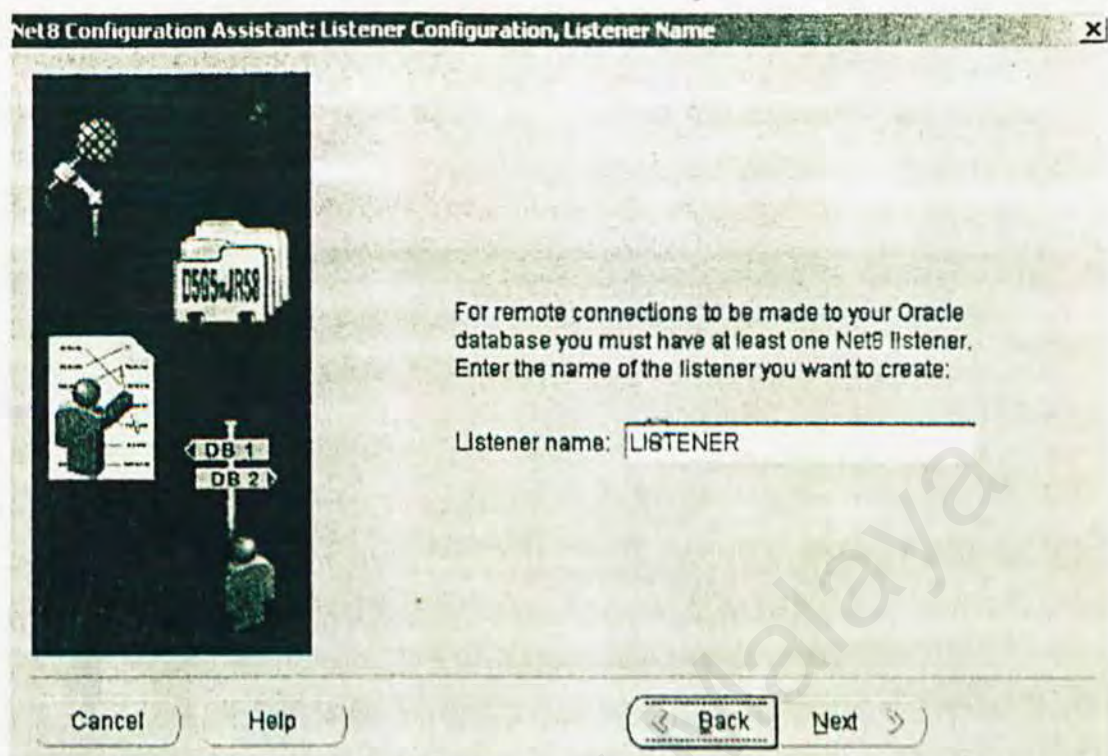


Figure 9.14: Naming the listener

The next step is to choose the protocol for this listener.

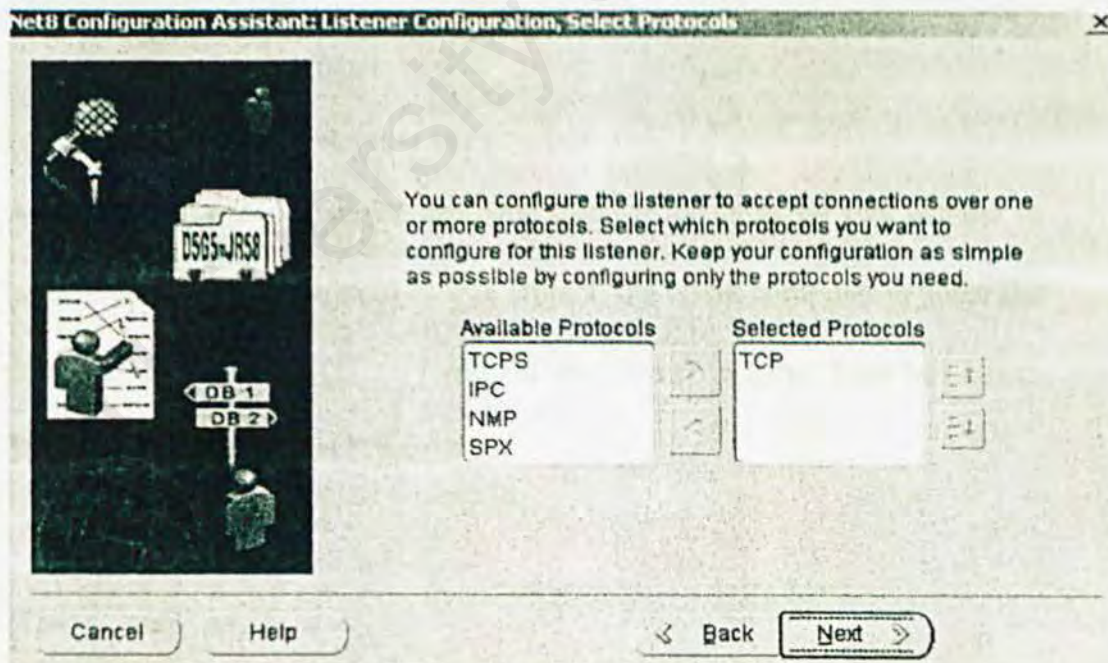


Figure 9.15: Select protocol

Lastly, select a standard port for the listener to use. For this, we always use the standard or default value of the port number recommended that is port number 1521.

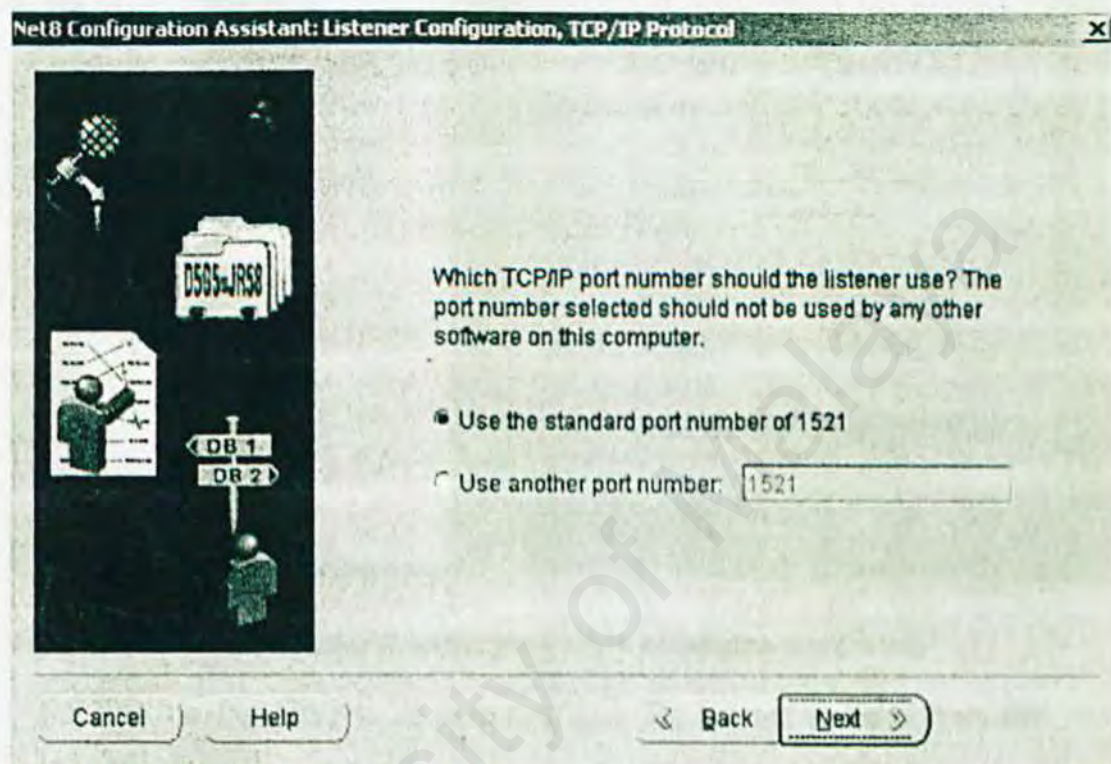


Figure 9.16: Select port number 1521

After creating a new listener, the next thing to do is to set this listener to listen from a specific client. For this project, the client computer is under the name of "CHEEKEONG", so I will set the listener to listen from this client. The example is shown in Figure 9.17.

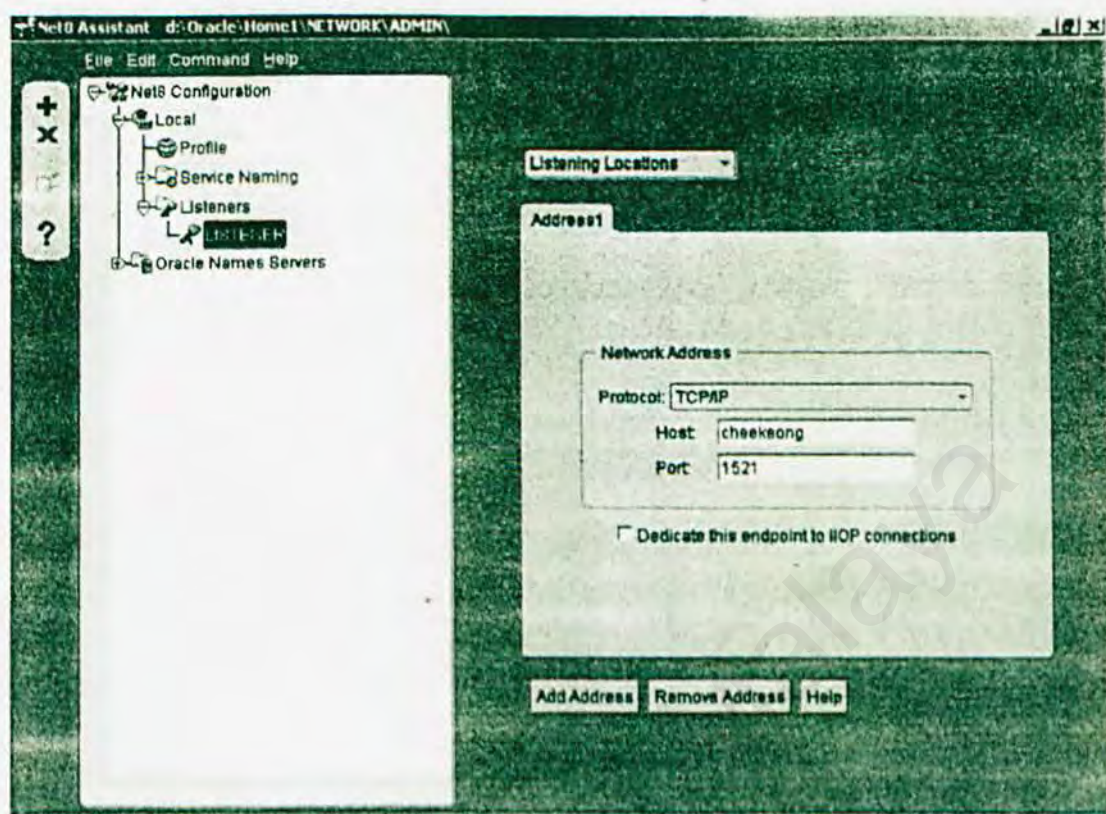


Figure 9.17: Set the listener to listen from client computer, cheekeong

After that, the last thing to do is to configure the listener to listen from the service name, FSKTM that I have previously created when I create the Local Naming method. The example is shown in Figure 9.18.



Figure 9.18: Set the listener to listen from service name, FSKTM

After all configurations are ready, we have to test the connection to make sure that the Oracle Server can listen from the Oracle Client without any faults. The user should receive a message such as the one in Figure 9.19 if the connection is correctly built.

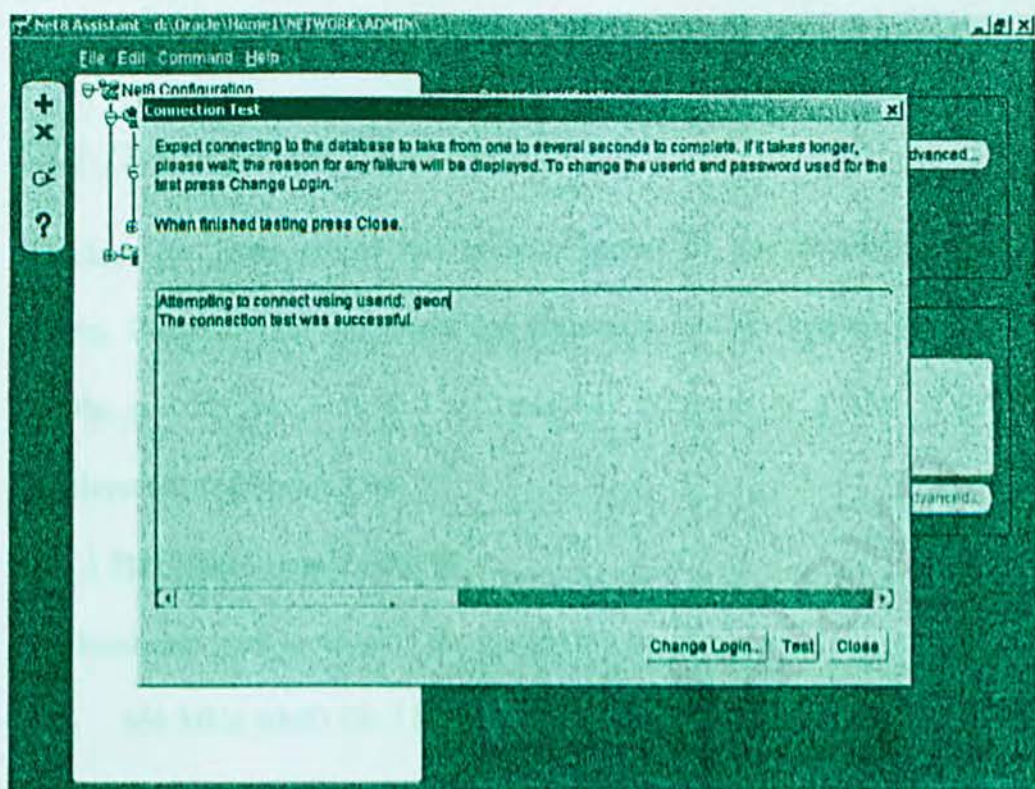


Figure 9.19: Testing the connection between Oracle Server and Oracle Client

Until this phase, the configuration of NET8 was successfully done and has been test for its connectivity. So I can say that the configuration of the Oracle Server and Oracle Client have been also successfully done correctly. The process of configuring the server and client is not too difficult, if to compare with the process of understanding the notes given on how to configure the client and the server. But after one can understand the notes, it will be sure easy for him to configure the connection because all of the configurations were in graphical user interface. In the next chapter, I will discuss on how to make a connection string from Visual Basic to Oracle Server.

Chapter 10 Implementation and Testing

10.1 Development Environment

Development environment has certain impact on the development of a system. Programming languages are important tool for system developing, but the suitable hardware and software will enhance the quality of system development and speed it up.

10.1.1 Hardware requirements

The hardware used to develop the system are as listed:

- 466 MHz AMD K6-2 Processor
- 192 MB SDRAM
- 14" Monitor
- Other standard desktop PC components

10.1.2 Tools For System Development

The tool used in the development phase is the Oracle Objects for OLE (OO4O). It is a middleware product manufactured by Oracle Corporation that allows native access (no ODBC) to Oracle databases from client applications via Microsoft OLE (Object Linking and Embedding) and COM. Oracle Objects is consists of an OLE 2.0 Automation (InProcess) Server - This provides an OLE Automation interface to applications that support OLE automation scripting such as Visual Basic (VB). The OO4O is installed automatically when the Oracle Client is installed to a particular computer.

10.1.3 Software Tools For System Development

Software	Usage
Microsoft 2000 Professional	As Operating System
Microsoft Visual Basic 6.0	System design and coding
Notepad	Coding
Oracle Client	Establishing Connection

10.1.4 Software Tools For Documentation

Microsoft Word 2000 is used to write the report because of its wide availability and user friendliness. Beside that, Microsoft Excel also has been used in some of the part in previous chapters for building Gantt chart.

10.2 Program Development

Program development is the process of creating the programs needed to satisfy an information system's processing requirements.

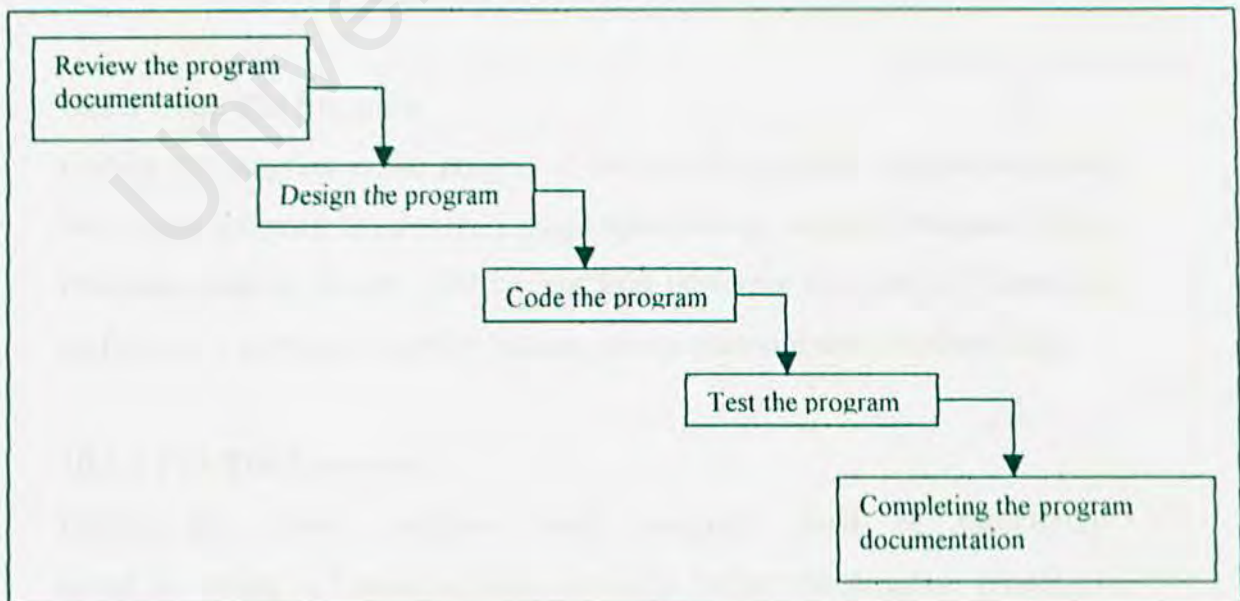


Figure 10.1: The five steps of program development

Program development consists of the following 5 steps: review the program documentation, design the program, code the program, test the program and completing the program documentation. (Figure 10.1)

10.2.1 Review The Program Documentation

The first step in the program development is to review the program documentation (report for WXES 3181) that was prepared during the previous phases. By reviewing the documentation, we can get better understanding on what will be done in implementation and testing phases. Also, documentation reminds us to stick on scope so that our goals and objectives will be achieved.

10.2.2 Design The Program

After the program documentation review, the second level that is the program design must be done during the system development cycle. For this second level of program design, questions like how the program can accomplish, what must program do to solve programming problems by logical solution are answered. The logical solution, or logic, for a program is a step-by-step solution to programming problems.

10.2.3 Code The Program

Coding the program is the process of writing the program instructions that implement the program design. Design specification must be translated into machine-readable format. The coding step performs this task. If design is performed in a detailed manner, coding can be accomplished mechanically.

10.2.4 Test The Program

During the testing program level, program must be thoroughly tested to ensure its functions work correctly before the program processes actual data and produces information on which people will rely on. In order

to do so, several types of test on an individual program have been done. The types of testing will be further discussed in details in the following section.

10.2.5 Document The Program

Accurate and complete program documentation is essential for the successful operations and maintenance of the information system. This documentation includes the system user manual that may be needed by most of the customers as well as the system administrator.

10.3 Program Coding

10.3.1 Coding Approach

A program with the technique called top-down, stepwise refinement, an approach that is essential to development of the well-structured program. This approach enables the programmer terminates the top-down, stepwise refinement process when the pseudocode algorithm is specified.

10.3.1.1 Coding style

Coding documentation is an important attribute of source code and it determines the intelligibility of a program. An easy to read source code makes the system easier to be maintained and enhanced. The elements of coding style include internal (source code level) documentation, methods for data declaration and approach to statement.

10.3.1.2 Code Documentation

Code documentation begins with the selection of identifier (variable and labels) names, continues with the composition of connectivity and end with the organization of the program. Use blank line or identification so that comments can be readily distinguished from code.

Internal Documentation – Internal comments provide a clear guide during maintenance phase of the system. Comments provide the developer with the means of communicating with other readers of the source code. A statement of purpose indicating the function of the module and descriptive comment that is embedded within the body of the source code is needed to describe processing functions.

Naming convention – Naming convention provides easy identification for the programmer. The naming convention as created with coding consistency and standardization in mind.*

Modularity – In order to reduce complexity; facilitate change results in easier implementation by encouraging parallel development of different parts of a system.

10.3.2 Database Connection

As I have stated earlier in this chapter, I have been using the Oracle Object for OLE to make a connection to the database. The reason why I am using the OO4O is because OO4O provides native access to Oracle and only Oracle databases, and is thus faster than ODBC access. ODBC is more generic and not database specific at all. OO4O closely follows the ODBC query interface model; one can retain over 95% code compatibility between the OO4O and ODBC versions of data layer procedures. The following table shows the objects available for use in OO4O.

OraSession	The first top-level object needed before we can connect to an Oracle database.
OraServer	Represents a physical connection to an Oracle database server instance. The OpenDatabase function

	can be used to create client sessions by returning an OraDatabase object.
OraDatabase	Represents a single login to an Oracle database. Similar to the ADO Connection object. OraDatabase objects are returned by the OraSession.OpenDatabase function.
OraConnection	Returns various pieces of user information about the current OraSession object. Many OraDatabase objects can share it, but each OraDatabase must exist in the same OraSession object.
OraDynaset	Similar to an ADO Recordset object. Represents the results retrieved by a call to the OraDatabase.CreateDynaset function.
OraField	Represents a column of data within an OraDynaset object. Similar to the ADO Field object of an ADO Recordset.
OraClient	Automatically created by OO4O as needed. Maintains a list of all active OraSession objects currently running on the workstation.
OraParameter	Represents a bind variable for a SQL statement or PL/SQL block to be executed using the OraDynaset object. Similar to the Parameter object in an ADO Command object.
OraParamArray	Allows arrays of parameters to be set for the OraDatabase.Parameters function.
OraSQLStmt	Represents a single SQL statement. Typically used with SQL statements that include bind variables to improve performance, as Oracle does not have to parse the statement each time it is executed. Can be thought of as conceptually similar to the ADO Command object.
OraMetaData	Returns meta data to describe a particular schema such as column names. Similar to the SQL Server DMO object library. See the meta data example below.
OraAQ	The CreateAQ method of the OraDatabase returns an OraAQ object. This provides access to Oracle's Advanced Queuing message system that allows messages to be passed between applications, much like MSMQ.

10.3.2.1 Connection Object

Here is the first piece of code that is used to create an OO4O connection Object:

```
Set OraSession = CreateObject("OracleInProcServer.XOraSession")
```

To build the connection object to the data store via OO4O, we supply the relevant connection information – such as the name of the Object, the method in the Object, and the parameters needed for this piece of code.

The next step is to open the specific database. Here we will use the code below:

```
Set Oradatabase = Orasession.OpenDatabase("fsktm", "geon/geonfei790520", 0&)
```

In the above code, we used the Opendatabase method to open the database by setting the parameters needed by this syntax. The parameters needed were, the database name, username, password and the opening type.

10.3.2.2 Retrieving Data Via OraDynaset

As stated in the table, the OraDynaset function is to represents the results retrieved by a call to the OraDatabase. The OraDynaset Object is used through its CreateDynaset method. One has to use this function before one can retrieve the data from a specific database. The syntax of using this method is shown below:

```
Set Oradyn = Oradb.CreateDynaset("select * from images", 0&)
```

10.3.2.3 Using The OraSql Stmt

This Object is used to represent a single SQL statement in OO4O, so that we are able to manipulate the data in an Oracle Database. Typically used with SQL statements that include bind variables to improve performance, as Oracle does not have to parse the statement each time it is executed. An example of the usage of this Object is shown below:

```
oradynaset.SQL = SQL_statement
```


10.3.2.4 OraBFile

From all the Objects that were used in program, the OraBFile is the most important Object that makes this application a successful one. Because our objective of doing this program is to produce an application that can retrieve multimedia data from an Oracle Database, so it is important that this Object exists in OO4O. I am saying this because, the OraBFile interface in OO4O provides methods for performing operations on large objects BFILE data type in the database. BFILEs are large binary data objects stored in operating system files (external) outside of the database table spaces. The methods exist in OraBFile Objects are Close, CloseAll, Open, CopyToFile, Clone, Compare, MatchPos and Read. The functions of these methods are as follows:

Close - Closes an opened BFILE.

CloseAll – This method closes all open OraBfile's on this connection.

Open – Open an OraBFILE.

CopyToFile - Copies a portion or all of the internal LOB value of this object to the local file.

Clone - Returns the Clone of OraLOB or OraBFILE object.

Compare - Compares the specified portion of the LOB value of an OraBlob or OraClob object (or OraBfile object) to the LOB value of the input OraBlob or OraClob object (or OraBfile object).

MatchPos - Returns the position of the nth occurrence of the pattern starting at the offset.

Read - Reads into a buffer a specified portion of BLOB, CLOB, or BFILE value. Returns the total amount of data read.

In the next page, I have put in two examples on how to access, read and insert a BFile.

Accessing BFile Value

BFILE data can be read using Read method. OraBFile allows piecewise read operation. Before reading the BFILE content, BFILE file should be opened using Open method. The coding is shown below:

```
Dim PartColl as OraBFile
```

```
Dim buffer As Variant
```

```
'Create a Dynaset containing a BLOB and a CLOB column
```

```
set part = OraDatabase.CreateDynaset ("select * from part",0)
```

```
PartColl = part.Fields("part_collateral").Value
```

```
'Open the bfile for read operation
```

```
PartColl.Open
```

```
'Read the entire bfile
```

```
amount_read = PartColl.Read(buffer)
```

```
'Close the bfile
```

```
PartColl.Close
```

Reading and Inserting BFile using Dynaset

Directory and filename of the BFILE value of OraBFile can be modified to new value by using DirectoryName and FileName properties. Lock should be obtained before modifying DirectoryName and FileName properties. For inserting new row containing BFILE column, the BFILE column must be initialized with new directory and file name value using DirectoryName and FileName properties. The coding is shown in next page.

Dim PartColl as OraBFile

Dim buffer As Variant

'Create a Dynaset containing a BLOB and a CLOB column

set part = OraDatabase.CreateDynaset ("select * from part",0)

PartColl = part.Fields("part_collateral").Value

'Insert a new BFILE in the part_collateral column

part.AddNew

'Directory objects will be upper-case by default

PartColl.DirectoryName = "NEWDIRECTORYNAME"

PartColl.FileName = "NewPartCollateral"

part.Update

'Move to the newly added row

part.MoveLast

'Open the Bfile for read operation

PartColl.Open

'Read the entire bfile

amount_read = PartColl.Read(buffer)

'Close the Bfile

PartColl.Close

10.4 System Testing

Testing is a critical element of software quality and represents the ultimate review of the specification, design and coding. Rules that serve well as testing objectives are:

- Testing is a process of executing a program with the intent of finding errors.
- A good test case is one that has probability of finding an undiscovered error.
- A successful test is one that uncovers and as yet undiscovered errors.

The system has undergone 3 stages of testing. They are the unit testing, integrating testing and system testing as shown in Figure 10.2. In Figure 10.2, the arrows from the top of the boxes indicate the normal sequence of testing. The arrows returning to the previous box indicate that previous testing stages may have to be repeated because of some problems. The stages in the testing process are:

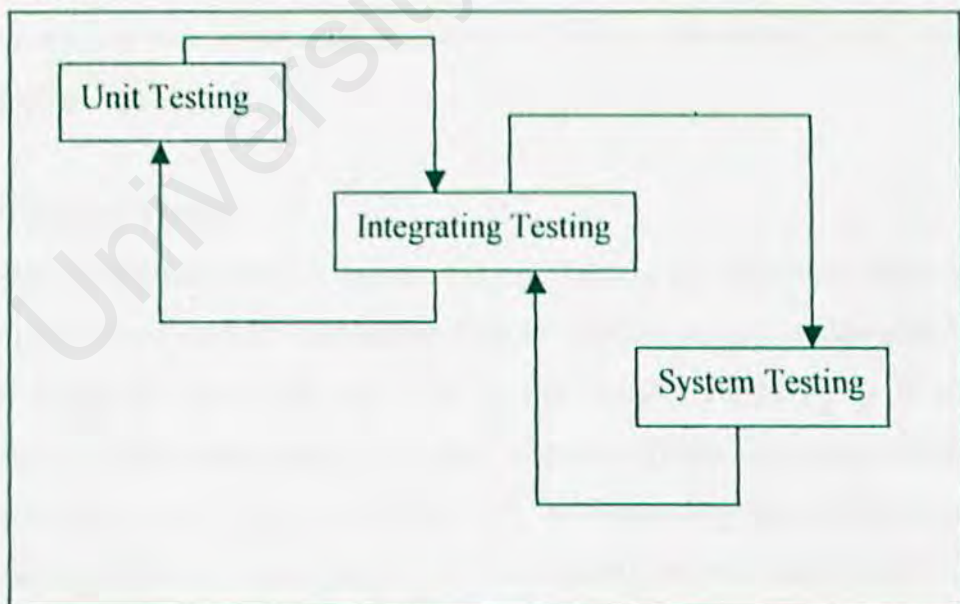


Figure 10.2 Testing Stages

10.4.1 Unit Testing

Historically, quality software is relied on testing each function or module. This practice is called unit testing, which is extremely time-consuming. For this system, unit was done during coding phase. The first step is to examine the program code by reading through it, spot the algorithm, data and syntax faults. Then comparing the code with specifications and with the design to make sure that all relevant cases have been considered. Finally, test cases are developed to show that input is properly converted to the desired output.

10.4.2 Integrating Testing

Testing a specific feature together with other newly developed features is known as integrating testing. In other words, when the individual components are working correctly, and meet the objectives, these components are combined into working system. Testing the interface of 2 components explores how components interact with each other. Incremental integration approach was applied during the developments of the system. The system was constructed and tested in small arguments, where errors were easier to isolate and correct. Error will be corrected before processing to the next integration.

10.4.3 System Testing

The last testing procedure is system testing. Testing the system is different from unit testing and the integration testing. System testing is designed to reveal bugs that cannot be attributed to individual component, or to the interaction among components and other objects. System tests study all the concerns issue and behaviors that can only be exposed by testing the entire integrated system or major part of it. Almost all system undergone three types of testing:

1. **Security Testing** – Verify the protection mechanism in the system against improper penetration.

2. **Stress Testing** – Stress test is to determine whether a program fulfill the requirements defined for it. Equally important is to make sure that program works, as it should, even under extreme condition.
3. **Performance Testing** – It is designed to test the run-time performance of system within the context of an integrated system. It occurs through all steps in the testing process.

Chapter 11 System Evaluation

This chapter states all the problems faced throughout the development of FSKTM Multimedia Database application. A list of the strengths and limitations of this application are also presented.

11.1 Problems Faced and Solutions

Various problems were encountered throughout the development of the FSKTM Multimedia Database application. The problems and the approaches taken to solve them are documented in the following sections.

11.1.1 Lack Of Knowledge In Establishing Connection

This is the major problem faced, as I have never been taught on how to connect two computers together in a client-server environment. What I have been taught are only theories about establishing connection but not the practical.

So what I have done was surfing the Internet to look for help and information about establishing client-server on Oracle. I have been also doing a lot of studies about the Oracle client-server establishment by borrowing books from the library and also by approaching some outsiders who are already have working experiences.

11.1.2 Lack Of Knowledge In Programming Language

The next problem that makes me become even more worried is about using the programming language. This is because; there are only a few numbers of examples and information on how to retrieve data from Oracle database through OO4O (Oracle Object for OLE) via Microsoft Visual Basic 6.0.

The only solution to this problem was to approach some IT personnel from various companies who have been using the OO4O and have deep knowledge about it.

Chapter 11 System Evaluation

This chapter states all the problems faced throughout the development of FSKTM Multimedia Database application. A list of the strengths and limitations of this application are also presented.

11.1 Problems Faced and Solutions

Various problems were encountered throughout the development of the FSKTM Multimedia Database application. The problems and the approaches taken to solve them are documented in the following sections.

11.1.1 Lack Of Knowledge In Establishing Connection

This is the major problem faced, as I have never been taught on how to connect two computers together in a client-server environment. What I have been taught are only theories about establishing connection but not the practical.

So what I have done was surfing the Internet to look for help and information about establishing client-server on Oracle. I have been also doing a lot of studies about the Oracle client-server establishment by borrowing books from the library and also by approaching some outsiders who are already have working experiences.

11.1.2 Lack Of Knowledge In Programming Language

The next problem that makes me become even more worried is about using the programming language. This is because; there are only a few numbers of examples and information on how to retrieve data from Oracle database through OO4O (Oracle Object for OLE) via Microsoft Visual Basic 6.0.

The only solution to this problem was to approach some IT personnel from various companies who have been using the OO4O and have deep knowledge about it.

11.1.3 Wide Area Of Studies

This application involves multimedia data types and 4 entities, which are the student, staff, facility and activity. So it is quite a big scope if to be compared with others student project. With a limited time, it is quite tuff to complete this project. But luckily, I still managed to finish this project on time.

To overcome this problem, a good management of time is very important and critical.

11.2 Strengths Of The System

11.2.1 Simple And Ease Of Use Graphical User interface

This application guarantees a simple and ease of use graphical user interface. This directly brings a user-friendly environment to the user and makes the manipulation of the database become so easy and convenient. It's all only about clicking buttons and entering short data. With the existence of combo boxes it's become even more convenient for the user to select their choice.

11.2.2 System Transparency

All the user need to do is just to click their queries or to type the data into the data field if they are adding some records to the database through the application. No intervention or any SQL statements are needed from the user.

11.2.3 Allow Searching Of Data

This application allows the user to search for the multimedia data types corresponding to a particular record.

11.2.4 Allow Sorting Of Record By Entity Type

The application allows the user to view the record by different entity types. A user is able to choose either to view the record of student, staff, activity or facility.

11.3 System Limitations

The following list the limitations of FSKTM Multimedia Database application.

11.3.1 Disadvantages Of Using OraBFile

One obvious disadvantage that we can discover from the usage of OraBFile is that, it doesn't support any deletion method. So we can only insert BFile into the database without having methods to delete it. The only way to delete the data is through manually done method. Which means that we have to go to the directory that stores the BFile and delete the BFile from there. After that by using SQL statement, manually delete the row that point to that deleted BFile from a table.

11.3.2 Limited Data Format That Can Be Read

This application can only read the image of the format JPEG and for audio file it can only read the audio file of the data format WAV. For video file, this application can only read an AVI format video file.

11.4 System Enhancement

The following list all the future enhancements that can be made to the FSKTM Multimedia Database application.

11.4.1 Change OraBFile to OraBlob or OraClob

Because the OraBlob and OraClob are able to put the Multimedia data type straight into the database table, so we will be able to delete the file in a more works saving manner, by means of using the SQL statement. So we need not have to do the deletion in a manually done way.

11.4.2 Login Form

In this application, I am assuming that the user of this application is the one that have full authority. So, I have set the default user login name and password to be the same as the administrator login name and password. For future, if this application is to be used by others partially authorized user, we can make a login form for them to enter their username and password. Then we can limit their scope of view by studying their username and password.

11.4.3 Integrate Plug-Ins Into Visual Basic

By integrating plug-ins, then this application will be able to read various data format files. Then, this application will be more powerful.

Chapter 12 User Manual for FSKTM Multimedia Database

It is always important to prepare a good user manual, because it will be the only reference for the user of that specific application. Because of this, I guarantee this user manual will be an interesting one and hope it is useful for my application user. So in this section, I will try my best to explain on how to use my application as well as giving guidance to those who will be going to use my application. Actually my application is fairly a user-friendly application although it does contain quite a number of forms. But it's only because; my application covers a big scope of database. So don't be worry!

12.1 Getting started: The First Page



Figure 12.1: FSKTM Multimedia Database Main Menu

Let's start with the most basic buttons, that an application should have that is the button of "FIRST", "LAST", "NEXT", "PREVIOUS", "ADD", "DELETE", "UPDATE" and "CANCEL". The function of these buttons is stated below:

FIRST – Move to the first record of the database.

LAST – Move to the last record of the database.

PREVIOUS - Move to the previous record.

NEXT – Move to the next record.

ADD – To add new record into the database.

DELETE – To delete any record from the database.

UPDATE – To update the database after any addition or deletion of record.

CANCEL – To terminate the whole program.

The "BACK TO MAINMENU" button is only for the user to return to the FSKTM Multimedia Database main menu. All the buttons above is easy to use, as I have mentioned before this application is a user-friendly application. After looking at how to manipulate text data from a database, let us go on to see how to manipulate image, audio and video data type from our database. In order to manipulate image, audio and video data type, we will have to use the "VIEW IMAGE", "VIEW VIDEO", "LISTEN AUDIO", "ADD IMAGE", "ADD VIDEO", "ADD AUDIO", "SEARCH ID" and "ADD DOCS" button. I will go through these buttons one by one and slowly because it is quite confusing when we come to this part of manipulating image, audio and video data type. But just stay cool because, it is only a bit more works to be done for manipulating the first data but after that, the next manipulation will become as easy as a snap of finger-tip.

Before I go further, allow me to tell you the assumption I have made to my application. First, because this is a multimedia database, I am assuming that each record that is to be added doesn't matter if it is from the category of student, staff, activity or facility, it must at least have on data of the type

image, audio or video. Second, to make our work easy and with fewer errors, I have standardized the ID number range for each categories and data types. For example, the DOCS_ID for STUDENT and STAFF is in the range of 10000-19999, this is follow by ACTIVITY ID with the range of 60000-69999, FACILITY ID (90000-99999), IMAGE ID (30000-39999), VIDEO ID 50000-59999) and lastly the AUDIO ID (40000-49999). It might seem to be a bit fussy but this is to ensure that we don't make any mistake when trying to add data into the database. Before adding the image, video and audio data type, we have to create four directories namely "Activity", "Facility", "Student" and "Staff" in the directory d:\MMDOC. These directories are created to allow us to insert image, video and audio files into each of the category.

So let us start with adding the image data type. Click on the "ADD IMAGE" button on the form. Then a form as below will appears.

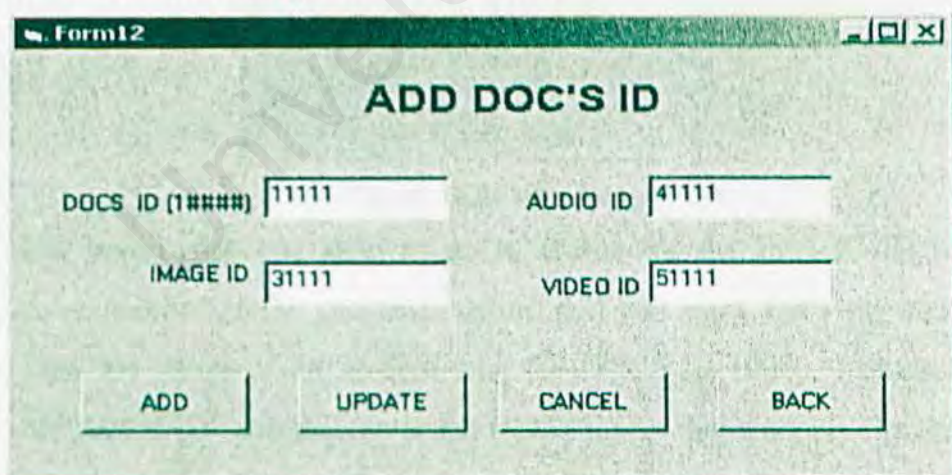


Figure 12.3: Add images form.

Enter the image id (remember it is of the range 30000-39999). Simply name the image name, size and the date created. When it comes to the filename field, please take note on this field. In this field you have to name the filename according to the name of the file that you have transferred into the specific directory. As for example, if you are trying to view a student image, you have to insert the student picture into the directory D:\MMDOC\Student. Then the filename will be the same as the filename that you have insert into D:\MMDOC\Student, follows by an extension of ". JPG". This concept goes

the same for audio and video file. Lastly, you have to choose the Directory alias from the directory field. If you are viewing student images, choose the "STUDENT_IMAGES" directory alias. As for others directory alias, it is chosen when you are viewing images from that particular category. The same happen when you are adding audio or video files. For instance, if you are adding a video file, let's say from a staff record; choose the directory as "STAFF_VIDEOS". After filling all the fields, click on UPDATE and then our job is done. To add on more images, just click on ADD, then the fields will all turn blank.

As I have stated in my first assumption on this application, one can only add in records about student, staff, activity or facility when the record has at least one DOCS_ID. Because DOCS table is a child to it's parents who are the IMAGE table, VIDEO table and AUDIO table, after we have initialised a new IMAGE ID, we have to also initialise a new DOCS ID for that particular IMAGE ID. We can only add in record about a category when a new DOCS ID has been created for that record. To add a DOCS_ID, just click on the "ADD DOCS" button and fills in the fields in the "ADD DOCS ID" form. A form as below will appears on the screen,



The screenshot shows a window titled "Form12" with a title bar containing standard window controls. The main content area is titled "ADD DOC'S ID" in bold. Below the title, there are four text input fields arranged in a 2x2 grid. The first field is labeled "DOCS ID (1#####)" and contains the text "11111". The second field is labeled "AUDIO ID" and contains "41111". The third field is labeled "IMAGE ID" and contains "31111". The fourth field is labeled "VIDEO ID" and contains "51111". At the bottom of the form, there are four rectangular buttons labeled "ADD", "UPDATE", "CANCEL", and "BACK" from left to right.

Figure 12.4: Add Doc's ID Form

The DOCS ID format will be of the same format of the IMAGE ID, as well as to the AUDIO ID and VIDEO ID, which means that if the IMAGE ID for one new added record is 31111 then the rest of the data type's ID will follow the last 4 digits from the IMAGE ID. This indicates that the VIDEO ID and AUDIO ID for this new added record will be 51111 and 41111 respectively. This is to make sure that there is no confusion between the new added records with the records that already exists. After everything is settled, we are all set to add a new record into the student, staff, activity or facility table. After discussing so much about adding the multimedia data types, let me explain to you on how to retrieve those data from the database. We take image data type for example. Click the "VIEW IMAGE" button, and you will see the form below.

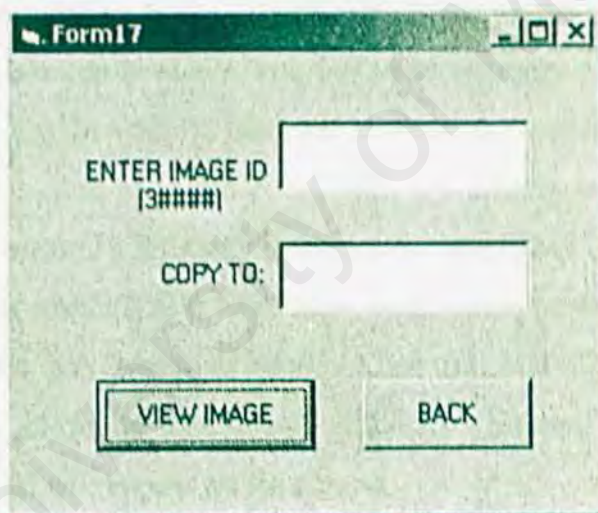
The image shows a screenshot of a software window titled "Form17". Inside the window, there are two text input fields. The first field is labeled "ENTER IMAGE ID (3#####)" and the second field is labeled "COPY TO:". Below these fields, there are two buttons: "VIEW IMAGE" and "BACK". The window has a standard Windows-style title bar with minimize, maximize, and close buttons.

Figure 12.5: View Image Form

But before this you have to go to search for the IMAGE ID, which is corresponding to the particular record that you want. Let's say you want to view the image of the student with the DOCS ID 10000. Just click on the "SEARCH ID" and then enter the DOCS ID for that student. The next things that are coming out will be the IMAGE ID, VIDEO ID and AUDIO ID corresponding to that student. This is show in the Figure 12.6.

Form1.4

SEARCH IDs

DOCS ID (1#####) 10000

AUDIO ID 40000

IMAGE ID 30000

VIDEO ID 50000

SEARCH BACK

Figure 12.6: Search Ids Form

From the “SEARCH IDs” form, you can get the IMAGE ID, VIDEO ID and AUDIO ID corresponding to the student who is having the DOCS ID 10000. Then from here, we can continue with our viewing image process. Enter the IMAGE ID into the IMAGE ID field and also specify to which location that this file will be copied to in the client side. Do enter a valid path! As for instance, I have created a directory E:\client\ for this file to be copied to. So I will simply enter the “COPY TO” field as E:\client\parameter.jpg, where parameter can be any string of name. After finished filling the “VIEW IMAGE” form, click on “DISPLAY IMAGE” button, a form namely DISPLAY IMAGE will appear on the screen.

The image shows a graphical user interface window titled "Form-4". The window has a title bar with standard minimize, maximize, and close buttons. The main content area is titled "DISPLAY IMAGE" in bold, centered text. Below the title is a large, empty rectangular box, likely intended for displaying an image. At the bottom of the window, there is a text input field preceded by the label "ENTER FILENAME :". To the right of the input field are two buttons: "DISPLAY" and "BACK".

Figure 12.7: Display Image Form

The filename field is filled with the name same as the parameter name when you filled the "COPY TO" field in the "VIEW IMAGE" form. After entering the filename, click on the "DISPLAY" button, and the image of that particular student entity will appear in the image box above.

After discussing about how to view the image data type, we will now go on with the audio and video data type. I will only explain on one of them because the processes of playing the audio or video data type will be the same all the way through. I will explain the audio data type here. If one wants to listen to the audio data type of a particular entity, as with viewing the image, one has to search for the AUDIO ID first. The method is the same, which means that one has to go to "SEARCH IDs" form to get the AUDIO ID. After knowing the AUDIO ID, click on the "LISTEN AUDIO" button and you will see the form as in Figure 12.8.

Form7

LISTEN AUDIO

ENTER AUDIO ID (4#####)

COPY TO:

Figure 12.8: Listen Audio Form

Just enter the corresponding AUDIO ID and specify to where the file should be copied to from the database. I have made a default path for this field that is E:\client\parameter.wav, where parameter is simply a string of name. Then just click on “PLAY AUDIO” and a new form as in Figure 12.9 will appears on the screen.

Form8

PLAY AUDIO

Figure 12.9: Play Audio Form

Click on the “Open Wave File” button. A Common Dialog box will appear for the user to browse to the specific path where the audio file has been copied. The Common Dialog box is shown as below.

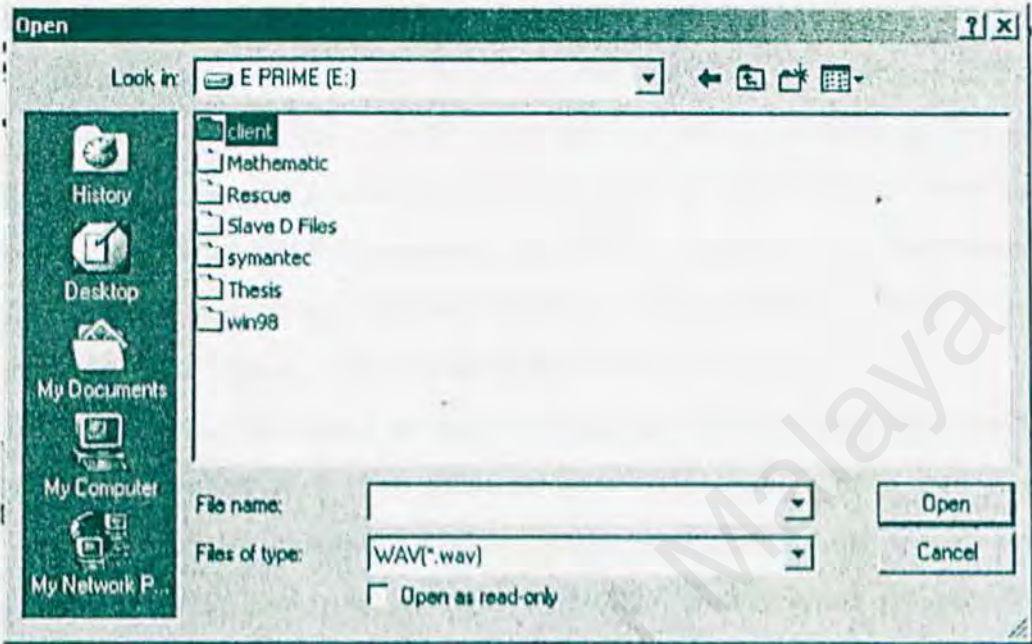


Figure12.10: Common Dialog Box

After choosing the specific path, click the “open” button in the Common Dialog box. Then, click the “play” button, which is the button with a triangle symbol on it, and then enjoy the output.

Until now, I have covered all about navigating, adding, deleting and retrieving the multimedia data type from the database. I really hope this user manual will help the first timers in using my application. Hope they will enjoy this application. Thanks to those who are willing to try this application. Thanks!!!

Chapter 13 Conclusion

Finally this report is ended with this section, the conclusion. Here, I will try to summarize our discussion.

In the first part, I have clearly stated the problem definition and how is the client-server architecture is better than others existing architectures. I have also given a lot of the client-server advantages in many kind of views in compare with the others architecture. Then later in the first part, I have stated the scope of this project and also building a Gantt chart to plan how my project will be done steadily according to the time I have.

In second part of this report, we have reviewed the literature that I have found out during my searching state. Many relevant materials have been reviewed. Firstly, some basic knowledge of client architecture, then, follows by network technologies and server technologies. Secondly, studies about the security features in client-server architecture and finally the analysis on existing system models.

For the third part, the methodology, I have discussed about the methods that I will be using and solution for my system development. I have also discussed on my system design, system module and system architecture in this particular part.

After discussing a lot of the theories, finally I come to the implementation part. In the implementation part, ways to establish client-server connection and also building an application called "FSKTM Multimedia Database" to retrieve data from the database server were stated. An interesting user manual then ends this implementation part.

Finally, this thesis has been finished. Although we have finished all the major task of the system, this system is still can be improved. I really hope that one-day this application will be enhanced and then being used by our faculty.

References:

Books

- [1] Coulouris G., Dollimore J. and Kindberg T. (2001). *Distributed System Concepts and Design, Third Edition*, England, Addison-Wesley, pp.16-17.
- [2] Metcalfe, R.M. and Boggs, D.R. (1976). *Ethernet: distributed packet switching for local computer networks*. Comms. ACM, Vol. 19, pp. 395-403.
- [3] Coulouris G., Dollimore J. and Kindberg T. (2001). *Distributed System Concepts and Design, Third Edition*, England, Addison-Wesley, pp.112.
- [4] Institute of Electrical and Electronic Engineers (1985). *Local Area Network- CSMA/CD Access Method and Physical Layer Specifications*. American National Standard ANSI/IEEE 802.3, IEEE Computer Society.
- [5] F. Kurose J. and W. Ross K. *Computer Networking : A Top-Down Approach Featuring the Internet*, Addison-Wesley, pp. 50.
- [6] F. Kurose J. and W. Ross K. *Computer Networking : A Top-Down Approach Featuring the Internet*, Addison-Wesley, pp. 427.
- [7] G. Page W., Jr., Austin D., Baird II W., Burke M., Chase N., Duer F., Gasper T., Hotka D., Kakade M., Lunawat V., F. Page B., Sharma P. and Thakkar M., *Special Edition Using Oracle8TM/8iTM*, USA, QUE, pp.420.

- [8] G. Page W., Jr., Austin D., Baird II W., Burke M., Chase N., Duer F., Gasper T., Hotka D., Kakade M., Lunawat V., F. Page B., Sharma P. and Thakkar M., *Special Edition Using Oracle8TM / 8iTM*, USA, QUE, pp.423.
- [9] Schneier, B. (1996). *Applied Cryptography, second edition*. New York: John Wiley.
- [10] Rivest, R.L., Shamir, A. and Adelman, L. (1978). *A method of obtaining digital signatures and public key cryptosystems*. Comms. ACM, Vol. 21, No. 2, pp.120-126.
- [11] Coulouris G., Dollimore J. and Kindberg T. (2001). *Distributed System Concepts and Design, Third Edition*, England, Addison-Wesley, pp. 282.
- [12] F. Kurose J. and W. Ross K. *Computer Networking : A Top-Down Approach Featuring the Internet*, Addison-Wesley, pp. 581.
- [13] Edwards J. and Devoe D. (1997), *3-tier Client Server At Work*, USA, John Wiley & Sons, Inc., pp.4-7.

Other References:

- I. Salemi J. (1995), *Guide To Client Server Databases, Second Edition*, Ziff-Davis Press, Emeryville California.
- II. Stallings, W. (1999), *Cryptography and Network Security- Principles and Practice, second edition*, Upper Saddle River, NJ: Prentice Hall.
- III. Comer, D.E. (1991) *Internetworking with TCP/IP, Volume 1: Principles, Protocols and Architecture, second edition*, Englewood Cliffs, NJ: Prentice-Hall.

World Wide Web

1. <http://www.npsinc.com/NPSi/papers/clientsv/clientsv.htm>

An Introduction to Client-Server Technology.

2. <http://pioneer.cc.edu/gli/Spring%2098/CSC409/chap11/team4/chapter11team4/sld009.htm>

About client/server and its architecture.

3. <http://docs.rinet.ru:8083/IntraBu/ch21.htm>

About how to build a client/server application.

4. <http://www.techsoup.org/articlepage.cfm?ArticleId=209&topicid=3>

About client/server networks.

5. http://www.wnt.gsi.de/oragsidoc/doc_804/network.804/a58230/toc.htm

About using net8 as a middleware in Oracle's client/server database.

6. <http://www.vbhow.to/books/vb6cs/fm/fm.asp>

About using visual basic for building a client/server application.